

# Базы данных

---

Рабочая тетрадь

Автор:  
Святослав Куликов

Данный документ предназначен **ТОЛЬКО** для использования студентами  
и преподавателями кафедры ПОИТ.

# Содержание

|   |           |
|---|-----------|
| <b>ПРЕДИСЛОВИЕ .....</b>                                      | <b>6</b>  |
| <b>ТЕМА 1: ОСНОВЫ БАЗ ДАННЫХ.....</b>                         | <b>7</b>  |
| <b>1.1. ДАННЫЕ И БАЗЫ ДАННЫХ .....</b>                        | <b>7</b>  |
| <b>1.2. МОДЕЛИ БАЗ ДАННЫХ.....</b>                            | <b>8</b>  |
| 1.2.1. Понятие модели базы данных.....                        | 8         |
| 1.2.2. Инфологические модели .....                            | 9         |
| 1.2.3. Даталогические модели .....                            | 10        |
| 1.2.4. Физические модели.....                                 | 11        |
| <b>1.3. ВИДЫ БАЗ ДАННЫХ .....</b>                             | <b>12</b> |
| <b>1.4. РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ .....</b>                   | <b>14</b> |
| 1.4.1. Определение .....                                      | 14        |
| 1.4.2. Основные факты о реляционной модели данных.....        | 14        |
| 1.4.3. Достоинства реляционной модели данных .....            | 15        |
| 1.4.4. Недостатки реляционной модели данных.....              | 16        |
| 1.4.5. Реляционные базы данных .....                          | 17        |
| <b>1.5. РЕЛЯЦИОННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ.....</b> | <b>18</b> |
| <b>1.6. ИСТОРИЯ РАЗВИТИЯ БАЗ ДАННЫХ .....</b>                 | <b>21</b> |
| <b>ТЕМА 2: ОТНОШЕНИЯ, КЛЮЧИ, СВЯЗИ.....</b>                   | <b>23</b> |
| <b>2.1. ОТНОШЕНИЯ.....</b>                                    | <b>23</b> |
| 2.1.1. Определение .....                                      | 23        |
| 2.1.2. Отношения в реляционной теории и в реальности .....    | 24        |
| <b>2.2. КЛЮЧИ.....</b>  | <b>25</b> |
| 2.2.1. Определение .....                                      | 25        |
| 2.2.2. Виды ключей .....                                      | 25        |
| <b>2.3. ИНДЕКСЫ.....</b>                                      | <b>32</b> |
| 2.3.1. Определение .....                                      | 32        |
| 2.3.2. Виды индексов .....                                    | 35        |
| 2.3.3. Управление индексами в MySQL .....                     | 37        |
| <b>2.4. СВЯЗИ .....</b>                                       | <b>38</b> |
| 2.4.1. Определение .....                                      | 38        |
| 2.4.2. Виды связей .....                                      | 38        |
| 2.4.3. Идентифицирующая и неидентифицирующая связи.....       | 41        |
| <b>2.5. ССЫЛОЧНАЯ ЦЕЛОСТНОСТЬ ДАННЫХ .....</b>                | <b>42</b> |
| 2.5.1. Определение .....                                      | 42        |
| 2.5.2. Каскадные операции .....                               | 42        |
| 2.5.3. Консистентность данных.....                            | 44        |
| <b>2.6. ТРИГГЕРЫ.....</b>                                     | <b>45</b> |
| <b>ТЕМА 3: НОРМАЛЬНЫЕ ФОРМЫ.....</b>                          | <b>46</b> |
| <b>3.1. АНОМАЛИИ ОПЕРАЦИЙ С БД.....</b>                       | <b>46</b> |
| 3.1.1. Важно помнить .....                                    | 46        |
| 3.1.2. Аномалии и их опасность.....                           | 46        |
| <b>3.2. ТЕОРИЯ ЗАВИСИМОСТЕЙ .....</b>                         | <b>49</b> |
| 3.2.1. Функциональная зависимость .....                       | 49        |
| 3.2.2. Избыточная функциональная зависимость .....            | 49        |

|  |  |            |
|--|--|------------|
| 3.2.3.   | Полная функциональная зависимость .....                            | 49         |
| 3.2.4.   | Частичная функциональная зависимость .....                         | 50         |
| 3.2.5.   | Транзитивная функциональная зависимость.....                       | 51         |
| 3.2.6.   | Многозначная зависимость .....                                     | 51         |
| 3.2.7.   | Тривиальная и нетривиальная многозначная зависимость.....          | 52         |
| <b>3.3.</b>  | <b>НОРМАЛИЗАЦИЯ И НОРМАЛЬНЫЕ ФОРМЫ.....</b>                        | <b>53</b>  |
| 3.3.1.   | ОПРЕДЕЛЕНИЕ .....  | 53         |
| 3.3.2.   | ТРЕБОВАНИЯ НОРМАЛИЗАЦИИ .....                                      | 53         |
| 3.3.3.   | НОРМАЛЬНЫЕ ФОРМЫ НИЗКИХ ПОРЯДКОВ .....                             | 56         |
| 3.3.4.   | НОРМАЛЬНЫЕ ФОРМЫ ВЫСОКИХ ПОРЯДКОВ .....                            | 59         |
| 3.3.5.   | КРАТКИЙ СПРАВОЧНИК ПО НОРМАЛЬНЫМ ФОРМАМ .....                      | 65         |
| 3.3.6.   | ПРИМЕР ПРИМЕНЕНИЯ НОРМАЛИЗАЦИИ .....                               | 66         |
| <b>3.4.</b>  | <b>ДЕНОРМАЛИЗАЦИЯ .....</b>  | <b>68</b>  |
| <b>ТЕМА 4: ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ.....</b>        |  | <b>69</b>  |
| <b>4.1.</b>  | <b>ОЦЕНКА СЛОЖНОСТИ БД.....</b>                                    | <b>69</b>  |
| 4.1.1.   | БЫСТРАЯ ОЦЕНКА.....  | 69         |
| 4.1.2.   | НАВЫКИ, НЕОБХОДИМЫЕ ДЛЯ УСПЕШНОГО ПРОЕКТИРОВАНИЯ БАЗЫ ДАННЫХ ..... | 71         |
| <b>4.2.</b>  | <b>ПРОЕКТИРОВАНИЕ БД НА ИНФОЛОГИЧЕСКОМ УРОВНЕ.....</b>             | <b>72</b>  |
| <b>4.3.</b>  | <b>ПРОЕКТИРОВАНИЕ БД НА ДАТАЛОГИЧЕСКОМ УРОВНЕ .....</b>            | <b>77</b>  |
| <b>4.4.</b>  | <b>ПРОЕКТИРОВАНИЕ БД НА ФИЗИЧЕСКОМ УРОВНЕ .....</b>                | <b>96</b>  |
| <b>4.5.</b>  | <b>ОСНОВЫ ИСПОЛЬЗОВАНИЯ SPARX EA .....</b>                         | <b>100</b> |
| 4.5.1.   | Создание модели и работа с инфологическим уровнем .....            | 100        |
| 4.5.2.   | Работа с даталогическим и физическим уровнем.....                  | 103        |
| 4.5.3.   | Создание скрипта генерации БД .....                                | 108        |
| <b>ТЕМА 5: ОСНОВЫ ЯЗЫКА SQL (НА БАЗЕ MYSQL).....</b> |  | <b>110</b> |
| <b>5.1.</b>  | <b>ОБЩИЕ СВЕДЕНИЯ ОБ SQL.....</b>                                  | <b>110</b> |
| <b>5.2.</b>  | <b>ИМЕНОВАНИЕ СТРУКТУР В MYSQL .....</b>                           | <b>111</b> |
| <b>5.3.</b>  | <b>ТИПЫ ДАННЫХ В MYSQL .....</b>                                   | <b>113</b> |
| <b>5.4.</b>  | <b>ЭЛЕМЕНТАРНОЕ УПРАВЛЕНИЕ ДАННЫМИ В MYSQL .....</b>               | <b>117</b> |
| 5.4.1.   | Выборка данных .....   | 117        |
| 5.4.2.   | Вставка данных .....   | 119        |
| 5.4.3.   | Удаление данных .....  | 119        |
| <b>5.5.</b>  | <b>ОСНОВЫ ИСПОЛЬЗОВАНИЯ MYSQL WORKBENCH .....</b>                  | <b>120</b> |
| <b>5.6.</b>  | <b>ОСНОВЫ ИСПОЛЬЗОВАНИЯ PHPMYADMIN .....</b>                       | <b>126</b> |
| <b>ТЕМА 6: ЯЗЫК УПРАВЛЕНИЯ ДАННЫМИ В MYSQL.....</b>  |  | <b>135</b> |
| <b>6.1.</b>  | <b>ОПЕРАТОР SELECT, ОБЩАЯ СТРУКТУРА.....</b>                       | <b>135</b> |
| 6.1.1.   | Обзор структуры и краткие пояснения .....                          | 135        |
| 6.1.2.   | SELECT: [ALL   DISTINCT   DISTINCTROW ] .....                      | 135        |
| 6.1.3.   | SELECT: [HIGH_PRIORITY].....                                       | 138        |
| 6.1.4.   | SELECT: [STRAIGHT_JOIN].....                                       | 140        |
| 6.1.5.   | SELECT: [SQL_SMALL_RESULT] [SQL_BIG_RESULT] .....                  | 141        |
| 6.1.6.   | SELECT: [SQL_BUFFER_RESULT] .....                                  | 144        |
| 6.1.7.   | SELECT: [SQL_CACHE   SQL_NO_CACHE] .....                           | 144        |
| 6.1.8.   | SELECT: [SQL_CALC_FOUND_ROWS].....                                 | 145        |
| 6.1.9.   | SELECT: SELECT_EXPR [, SELECT_EXPR ...] .....                      | 147        |
| 6.1.10.  | SELECT: FROM TABLE_REFERENCES .....                                | 152        |
| 6.1.11.  | SELECT: [WHERE WHERE_CONDITION] .....                              | 155        |
| 6.1.12.  | SELECT: [GROUP BY ...] .....                                       | 157        |
| 6.1.13.  | SELECT: [HAVING WHERE_CONDITION] .....                             | 161        |

|   |            |
|---|------------|
| 6.1.14. SELECT: [ORDER BY ...] .....  | 162        |
| 6.1.15. SELECT: [LIMIT ...] .....   | 165        |
| 6.1.16. SELECT: [PROCEDURE PROCEDURE_NAME(ARGUMENT_LIST)] .....   | 169        |
| 6.1.17. SELECT: [INTO OUTFILE ...] .....  | 170        |
| 6.1.18. SELECT: [INTO DUMPFILE ...] .....   | 171        |
| 6.1.19. SELECT: [INTO VAR_NAME [, VAR_NAME]]] .....   | 172        |
| 6.1.20. SELECT: [FOR UPDATE   LOCK IN SHARE MODE]] .....  | 173        |
| <b>6.2. «ПОДСКАЗКИ» ПО ИНДЕКСАМ .....</b>   | <b>175</b> |
| 6.2.1. Обзор структуры и краткие пояснения .....  | 175        |
| 6.2.2. INDEX_HINT: USE   IGNORE   FORCE .....   | 175        |
| 6.2.3. INDEX_HINT: [FOR {JOIN ORDER BY GROUP BY}] .....   | 177        |
| 6.2.4. Краткие итоги .....  | 178        |
| <b>6.3. ОБЪЕДИНЕНИЕ ЗАПРОСОВ (UNION) .....</b>  | <b>179</b> |
| 6.3.1. Обзор структуры и краткие пояснения .....  | 179        |
| 6.3.2. SELECT ... UNION .....   | 179        |
| 6.3.3. Пример использования UNION .....   | 180        |
| 6.3.4. Краткие итоги .....  | 181        |
| <b>6.4. ЗАПРОСЫ НА ОБЪЕДИНЕНИЕ (JOIN).....</b>  | <b>182</b> |
| 6.4.1. Назначение и общая структура JOIN .....  | 182        |
| 6.4.2. Принципы работы JOIN .....   | 184        |
| 6.4.3. Три простых примера использования JOIN .....   | 184        |
| 6.4.4. Основные разновидности JOIN .....  | 189        |
| 6.4.5. Все разновидности JOIN .....   | 191        |
| 6.4.6. Условия объединения и дополнительные условия .....   | 195        |
| 6.4.7. Все виды JOIN в примерах.....  | 197        |
| 6.4.8. JOIN'ы и NULL'ы.....   | 200        |
| 6.4.9. JOIN'ы и дублирование имён полей.....  | 202        |
| 6.4.10. Нетривиальные случаи, вопросы и примеры.....  | 203        |
| <b>6.5. ПОДЗАПРОСЫ (SUBQUERIES) .....</b>   | <b>209</b> |
| 6.5.1. Общие сведения о подзапросах .....   | 209        |
| 6.5.2. Скалярные подзапросы и сравнение с использованием подзапросов .....                                      | 209        |
| 6.5.3. Подзапросы с ключевыми словами ANY, IN, SOME, ALL .....  | 211        |
| 6.5.4. Подзапросы, возвращающие ряды .....  | 214        |
| 6.5.5. Подзапросы с ключевым словом [NOT] EXISTS .....  | 215        |
| 6.5.6. Взаимосвязанные запросы и подзапросы .....   | 216        |
| 6.5.7. Подзапросы как источник данных .....   | 216        |
| 6.5.8. Анализ ошибок в подзапросах, оптимизация подзапросов и преобразование подзапросов в запросы с JOIN ..... | 218        |
| 6.5.9. Промежуточный итог .....   | 220        |
| <b>6.6. ОПЕРАТОР UPDATE .....</b>   | <b>221</b> |
| 6.6.1. Общая структура оператора UPDATE .....   | 221        |
| 6.6.2. UPDATE: [LOW_PRIORITY] .....   | 221        |
| 6.6.3. UPDATE: [IGNORE].....  | 221        |
| 6.6.4. UPDATE: SET .....  | 222        |
| 6.6.5. UPDATE: [WHERE WHERE_CONDITION] .....  | 223        |
| 6.6.6. UPDATE: [ORDER BY ...] .....   | 223        |
| 6.6.7. UPDATE: [LIMIT ROW_COUNT].....   | 224        |
| 6.6.8. UPDATE: РАБОТА С НЕСКОЛЬКИМИ ТАБЛИЦАМИ .....   | 224        |
| <b>6.7. ОПЕРАТОР INSERT .....</b>   | <b>226</b> |
| 6.7.1. Общая структура оператора INSERT .....   | 226        |
| 6.7.2. Классический вариант синтаксиса INSERT .....   | 226        |
| 6.7.3. INSERT: DELAYED .....  | 227        |
| 6.7.4. INSERT: [INTO] .. ....   | 227        |
| 6.7.5. INSERT: ON DUPLICATE KEY UPDATE .....  | 228        |

|              |  |            |
|--------------|--|------------|
| 6.7.6.       | INSERT: [IGNORE] .....   | 229        |
| 6.7.7.       | Синтаксис INSERT в стиле UPDATE.....                           | 230        |
| 6.7.8.       | Синтаксис INSERT ... SELECT .....                              | 230        |
| 6.7.9.       | INSERT: ВАЖНОЕ НАПОМИНАНИЕ, ВОПРОСЫ И ОТВЕТЫ.....              | 232        |
| <b>6.8.</b>  | <b>ОПЕРАТОР REPLACE.....</b>                                   | <b>233</b> |
| 6.8.1.       | Общая структура и назначение оператора REPLACE .....           | 233        |
| 6.8.2.       | Области применения и пример использования REPLACE.....         | 233        |
| 6.8.3.       | Области применения и пример использования REPLACE.....         | 234        |
| 6.8.4.       | Краткий итог по REPLACE.....                                   | 234        |
| <b>6.9.</b>  | <b>ОПЕРАТОР DELETE .....</b>                                   | <b>235</b> |
| 6.9.1.       | Общая структура оператора DELETE.....                          | 235        |
| 6.9.2.       | DELETE: [QUICK].....   | 235        |
| 6.9.3.       | DELETE: [IGNORE] .....   | 235        |
| 6.9.4.       | DELETE: пример и важное напоминание .....                      | 236        |
| 6.9.5.       | DELETE: работа с несколькими таблицами.....                    | 236        |
| 6.9.6.       | Способ быстрой полной очистки таблицы .....                    | 236        |
| <b>6.10.</b> | <b>ОПЕРАТОР LOAD DATA INFILE .....</b>                         | <b>237</b> |
| 6.10.1.      | Общая структура оператора LOAD DATA INFILE.....                | 237        |
| 6.10.2.      | LOAD DATA INFILE: [CONCURRENT] .....                           | 237        |
| 6.10.3.      | LOAD DATA INFILE: [CONCURRENT] .....                           | 238        |
| 6.10.4.      | LOAD DATA INFILE: [REPLACE   IGNORE] .....                     | 238        |
| 6.10.5.      | LOAD DATA INFILE: [CHARACTER SET CHARSET_NAME] .....           | 238        |
| 6.10.6.      | LOAD DATA INFILE: [FIELDS TERMINATED ... ENCLOSED ...] .....   | 239        |
| 6.10.7.      | LOAD DATA INFILE: [LINES STARTING ... TERMINATED BY ...] ..... | 239        |
| 6.10.8.      | LOAD DATA INFILE: [IGNORE NUMBER {LINES ROWS}].....            | 240        |
| 6.10.9.      | LOAD DATA INFILE: [SET COL_NAME = EXPR].....                   | 240        |
| 6.10.10.     | LOAD DATA INFILE: пример использования .....                   | 241        |
| <b>6.11.</b> | <b>ОПЕРАТОР LOAD XML.....</b>                                  | <b>242</b> |
| 6.11.1.      | Общая структура оператора LOAD XML .....                       | 242        |
| 6.11.2.      | LOAD XML: пример использования .....                           | 242        |
| 6.11.3.      | LOAD XML: краткий итог .....                                   | 243        |
| <b>6.12.</b> | <b>ОПЕРАТОР CALL.....</b>                                      | <b>244</b> |
| 6.12.1.      | Общая структура оператора CALL.....                            | 244        |
| 6.12.2.      | Пример хранимой процедуры и использования оператора CALL ..... | 244        |
| <b>6.13.</b> | <b>ОПЕРАТОР DO.....</b>  | <b>247</b> |
| 6.13.1.      | Общая структура оператора DO .....                             | 247        |
| <b>6.14.</b> | <b>ОПЕРАТОР HANDLER.....</b>                                   | <b>247</b> |
| 6.14.1.      | Общая структура оператора HANDLER .....                        | 247        |
| 6.14.2.      | Пример использования HANDLER .....                             | 247        |
| 6.14.3.      | Краткий итог по оператору HANDLER .....                        | 248        |
| <b>6.15.</b> | <b>ИТОГ ПО ВСЕМ ВЫРАЖЕНИЯМ УПРАВЛЕНИЯ ДАННЫМИ .....</b>        | <b>248</b> |

## Предисловие

Что такое «рабочая тетрадь»? В отличие от книги в рабочей тетради можно и **НУЖНО** делать пометки и записи. Это ВАШ материал, в котором отражено ВСЁ, что будет показано на презентациях.

Здесь есть специальные места для записей (разлинованные участки), но вы также можете что-то обводить, подчёркивать и т.д.

Этот материал можно будет использовать на всех практических занятиях, а также для повторения изученного и просто как своего рода справочник или «то, что можно просто почитать».

Наш тренинг охватывает материал, для полноценного усвоения которого вам рекомендуется приложить максимум усилий в выполнении практических заданий, использовать все свои знания из области ИТ и активно пользоваться технической документацией (часть которой придётся искать самостоятельно, т.к. невозможно предсказать заранее, какие затруднения могут возникнуть).

Автор рабочей тетради неставил перед собой задачу скопировать официальную документацию по инструментальным средствам, методологиям и т.п., а потому каждая рассмотренная тема НЕ является «исчерпывающим руководством», после изучения которого вы сразу станете гуру. НО! Вы всегда можете обратиться к официальной документации. Не удивляйтесь, что некоторой информации не будет и там: многие вещи авторы считают «самоочевидными». Что делать? Спрашивать тренера и пользоваться поиском в Интернет.

Помните: задача тренинга и данной рабочей тетради – обозначить ключевые моменты, отталкиваясь от которых вы сможете нарабатывать свой личный опыт значительно быстрее, чем при «полностью самостоятельном» исследовании рассмотренных тем, но никакая рабочая тетрадь и ни один тренер не заменят ваш личный опыт, все условия для приобретения которого мы создадим.

Итого:

- Больше вопросов – любых! Обязательно ЗАПИСЫВАЙТЕ вопросы, которые возникли у вас дома при повторении материала. Тогда вы вспомните о них на занятии и сможете задать.
- Пишите комментарии – везде! Это ВАША тетрадь. Используйте её!
- Обращайтесь к дополнительным источникам в Интернет и в документации, поставляемой с программами, которые мы рассмотрим.
- Используйте любую возможность, чтобы закрепить изученное на практике.

Успехов!

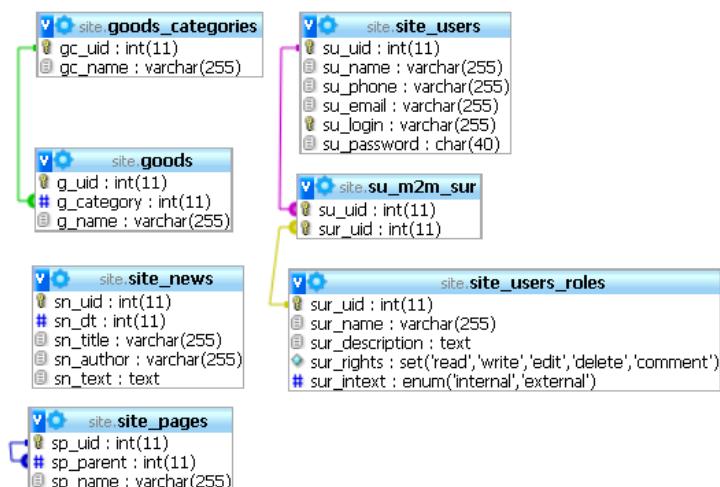
## Тема 1: Основы баз данных

### 1.1. Данные и базы данных

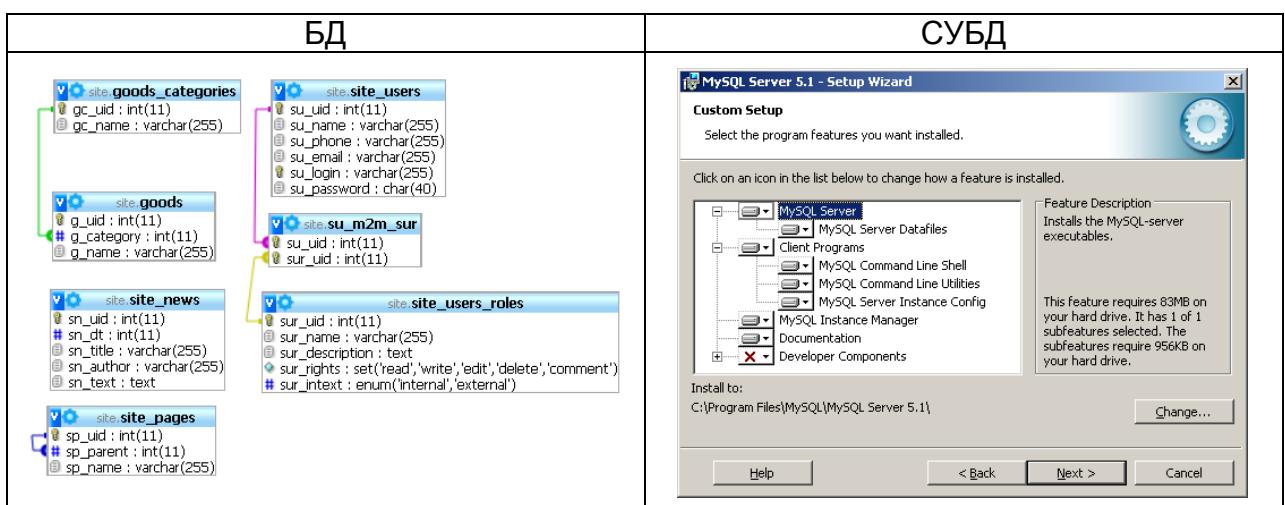
**Данные** (data) – представление фактов и идей в формализованном виде, пригодном для передачи и обработки в некотором информационном процессе.

| su_uid | su_name        | su_phone  | su_email        | E-mail<br>пользователя | su_login | su_password                              | Пароль пользователя |
|--------|----------------|-----------|-----------------|------------------------|----------|--|---------------------|
| 1      | Пупкин<br>В.В. | 123-45-67 | pupkin@mail.ru  |                        | pupkin   | 7c4a8d09ca3762af61e59520943dc26494f8941b |                     |
| 2      | John<br>Smith  | 111-22-33 | smith@gmail.com | smith                  | smith    | dd5fef9c1c1da1394d6d34b248c51be2ad740840 |                     |

**База данных** (БД, database) – структурированный организованный набор данных, описывающих характеристики какой-либо физической или виртуальной системы.



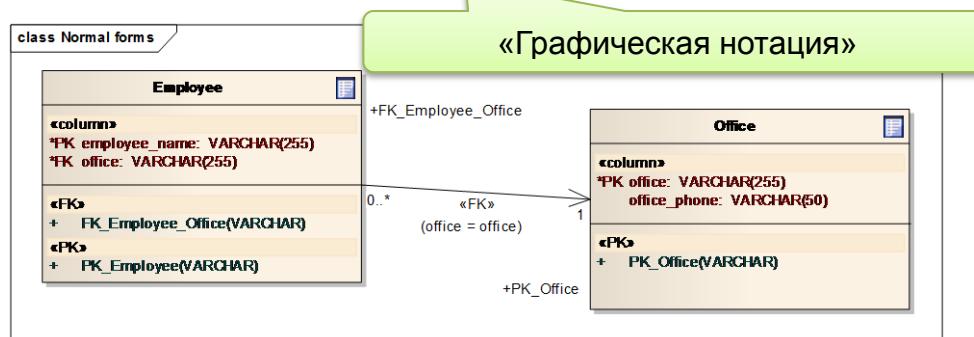
**Система управления базами данных** (СУБД, database management system) – программное обеспечение, предназначенное для организации и управления базами данных.



## 1.2. Модели баз данных

### 1.2.1. Понятие модели базы данных

**Модель базы данных** (database model) – описание базы данных с помощью определённого (в т.ч. графического) языка на некотором уровне абстракции.



### Уровни моделирования («уровни абстракции»)

| Уровень        | В чём суть  | Пример «из жизни»  |
|----------------|---|--|
| Инфологический | Описание предметной области, нет привязки к СУБД, предназначено «для людей».                | «Данные сотрудников будем хранить в личных делах».   |
| Даталогический | Модель предметной области в привязке к СУБД определённого вида или к конкретной СУБД.       | «Личное дело каждого сотрудника представляет собой два документа – листок по учёту кадров и биографию».              |
| Физический     | Таблицы, связи, индексы, методы хранения, настройки производительности, безопасности и т.п. | «Документы личного дела будут распечатаны на листах А4, сшиты в папку и спрятаны в сейф в углу кабинета начальника». |

Модель базы данных формируется на любом из уровней с учётом следующих требований:

| Адекватность предметной области  | Удобство использования  |
|--|---|
| <pre> class Basics class Employee {     &lt;&lt;column&gt;&gt;     name: VARCHAR(255)     birthdate: DATE     max_speed: BIGINT     max_cargo: BIGINT }   </pre> | <pre> SELECT distinct `dv`.device, `ov`.`os`, `dv`.`version` from `dv` join `do` on `dv`.`device` = `do`.`device` join `ov` on `do`.`os` = `ov`.`os`   </pre> <p>???      ???</p> |

## 1.2.2. Инфологические модели

На инфологическом уровне приводится описание предметной области без привязки к конкретной СУБД и в форме, предназначенной для удобного восприятия человеком.

Описывать модель здесь можно...

«Словами»  
(просто в виде списков)

«Табличками» (в Word или подобных редакторах)

Специальными графическими нотациями (UML и т.п.)

Новости:

- Заголовок
- Автор
- Дата публикации
- Текст

Новости

| Заголовок | Автор | Дата публикации | Текст |
|-----------|-------|-----------------|-------|
| Текст     | Текст | Дата            | Текст |

class Infological Model

```
class Infological Model
  News
    - Author: char
    - Header: char
    - Publishing Date: char
    - Text: char
```

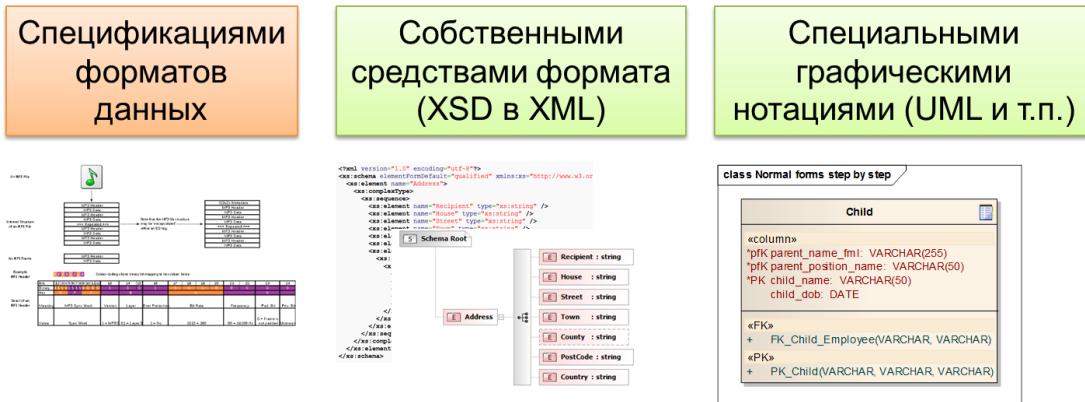
### Виды инфологических моделей

| Вид              | Пояснение   | Пример   |
|------------------|---|--|
| Семантическая    | Часто строится с использованием семантических сетей, показывает взаимосвязь объектов, понятий и т.п.              | <pre> graph TD     Кот -- Ест --&gt; Рыба     Кот -- Не любит --&gt; Вода     Вода -- Поливает --&gt; Растение     Рыба -- Живёт в --&gt; Вода   </pre>  |
| Графовая         | Полностью основана на теории графов, достаточно эффективна лишь в специфических предметных областях.              | <pre> graph LR     Абитуриент --&gt; 1[1-й курс]     1 --&gt; 2[2-й курс]     2 --&gt; 3[3-й курс]     3 --&gt; 4[4-й курс]     4 --&gt; 5[5-й курс]     5 --&gt; Выпускник     Отчислен --&gt; Выпускник     Выпускник --&gt; Абитуриент   </pre> |
| «Сущность-связь» | Идеально подходит для моделирования реляционных баз данных. Основана на понятиях «сущности», «атрибута», «связи». | <pre> class Infological Model   News     - Author: char     - Header: char     - Publishing Date: char     - Text: char   News Rubric     - Name: char   News &lt;--&gt; News Rubric   </pre>  |

### 1.2.3. Даталогические модели

На даталогическом уровне модель предметной области представляется в привязке к СУБД определённого вида или к конкретной СУБД и описывает способ организации данных безотносительно их физического размещения.

Описывать модель здесь можно...



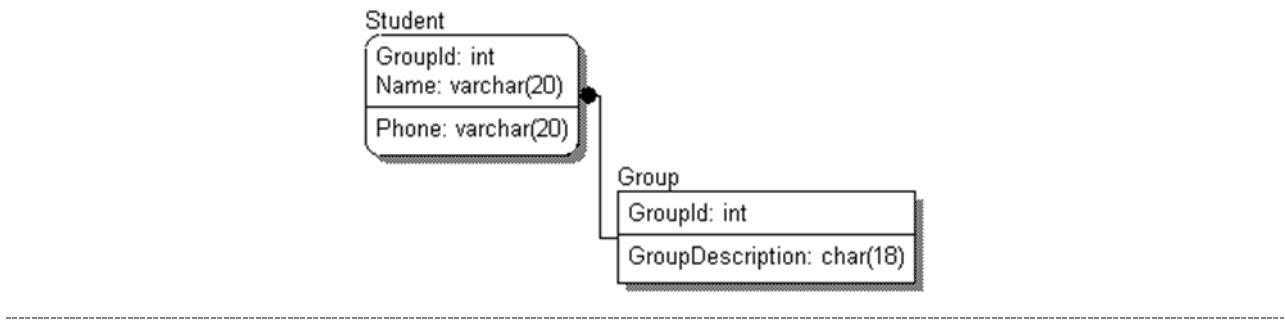
### Виды даталогических моделей (кратко, подробно будет позже)

| Модели                               | Пояснение   |
|--------------------------------------|---|
| Документальные (архивы)              | Обработка однотипных или разнотипных файлов по некоторым правилам         |
| Фактографические (карточки)          | Картотека («плоская таблица») в любом её виде                             |
| Теоретико-графовые                   | Сети и графы с соответствующими правилами обработки                       |
| Теоретико-множественные              | Например, <b>реляционная</b> («сущность-связь») – о ней речь пойдёт позже |
| Объектно-ориентированные             | См. ООП в программировании ☺  |
| Основанные на инвертированных файлах | Редкий случай. Лучший пример такого в жизни – «облачо тегов».             |

#### 1.2.4. Физические модели

Физический уровень описывает конкретные таблицы, связи, индексы, методы хранения, настройки производительности, безопасности и т.п.

Описывать модель здесь можно... как угодно. Чаще всего выбирают такой же способ представления, как и у даталогической модели. Из-за разнообразия предметных областей здесь используется всё – от чертежей и схем, до специальных нотаций (IDEF0).



### 1.3. Виды баз данных

Виды баз данных базируются на даталогических моделях и представлены следующим списком:

- Картотеки.
- Сетевые базы данных.
- Иерархические базы данных.
- Реляционные базы данных.
- Многомерные базы данных.
- Объектно-ориентированные базы данных.
- Дедуктивные базы данных.
- NoSQL базы данных.

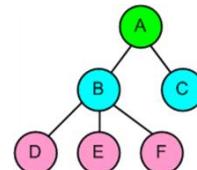
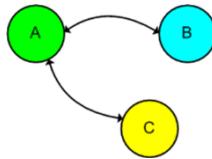
**Картотека** (card index) – упорядоченное (по алфавиту, дате и т.п.) собрание данных в виде записей («карт»), каждая из которых предоставляет сведения о каком-то объекте базы данных. Современный аналог картотек – «табличка» в Excel.

| Автор     | Книга               | Страниц |
|-----------|---------------------|---------|
| Азимов А. | Основание           | 154     |
| Азимов А. | Основание и империя | 102     |
| Азимов А. | Второе основание    | 104     |

В **сетевых** (network) БД каждый элемент может быть связан с другим, а **иерархические** (hierarchy) БД построены на основе той или иной иерархической структуры данных (например, на основе дерева).

Типичные примеры «иерархических БД»:

- Файловая система.
- Реестр Windows.
- LDAP и Active Directory.



**Реляционные** (relational) БД основаны на теоретико-множественной реляционной даталогической модели (предложена доктором Эдгаром Коддом в 1970 году). Все данные представлены в виде (связанных между собой) таблиц, разбитых на строки и столбцы.

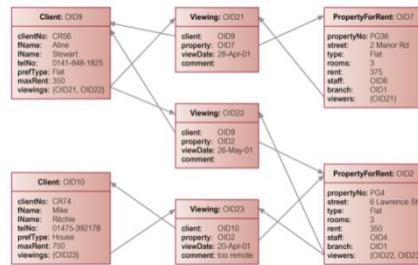
| bsu.users              |  | bsu.roles             |  | u_id |      | u_login                          |  | u_password |  | u_fio |         | u_role |          |
|------------------------|--|-----------------------|--|------|------|----------------------------------|--|------------|--|-------|---------|--------|----------|
| u_uid : int(11)        |  | r_uid : int(11)       |  | 1    | John | 61409aa1fd47d4a5332de23cbf59a36f |  | John Smith |  | 1     |         | ame    | r_rights |
| u_login : varchar(200) |  | r_name : varchar(200) |  | 2    | Joe  | 3a368818b7341d48660e8dd6c5a77dbe |  | Joe Black  |  | 2     | 1 Admin |        | 2 User   |

**Многомерные** БД (OLAP, online analytical processing) предназначены для обработки данных из различных источников и временных данных. Могут строиться на основе реляционных БД или на основе своих, более сложных хранилищ.



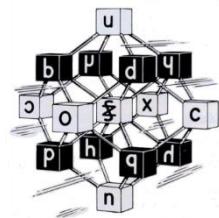
В **объектно-ориентированных** (object-oriented) БД данные оформлены в виде моделей объектов, включающих прикладные программы, которые управляются внешними событиями.

Эта технология напоминает объектно-ориентированное программирование (ООП) в применении к БД. СУБД для работы с такими технологиями бывают объектными и объектно-реляционными.



**Дедуктивная** (deductive) БД состоит из двух частей: экстенциональной (содержащей факты) и интенциональной (содержащей правила для логического вывода новых фактов).

Основным отличием дедуктивной СУБД от реляционной является то, что правила интенциональной части БД и запросы пользователей могут содержать рекурсию.



**NoSQL** (not only SQL) БД реализуют ряд подходов, имеющих существенные отличия от используемых в реляционных СУБД. Описание схемы данных в случае использования NoSQL-решений может осуществляться через использование различных структур данных: хеш-таблиц, деревьев и т.д.



## 1.4. Реляционная модель данных

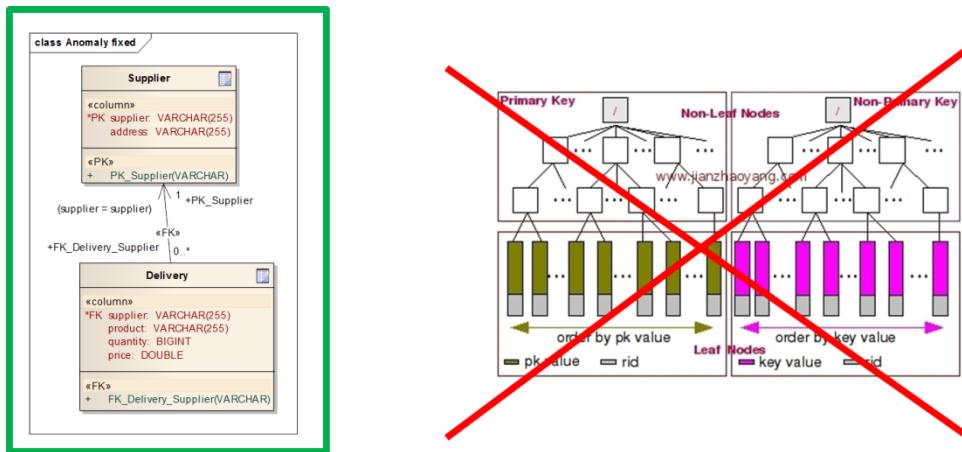
### 1.4.1. Определение

**Реляционная модель данных** (relational data model) – математическая теория, описывающая структурный аспект, аспект целостности и аспект обработки данных в реляционных базах данных.



### 1.4.2. Основные факты о реляционной модели данных

РМД является **логической**, т.е. отношения являются логическими (абстрактными), а не физическими (хранимыми) структурами.



Все данные представлены явным заданием значений атрибутов в кортежах отношений, т.е. нет никаких указателей (адресов), связывающих одно значение с другим.

| su_uid<br>Уникальный идентификатор пользователя | su_name<br>ФИО пользователя | su_phone<br>Телефон пользователя | su_email<br>пользователя | E-mail | su_login<br>Логин пользователя           | su_password<br>Пароль пользователя |
|---|-----------------------------|----------------------------------|--------------------------|--------|--|------------------------------------|
| 1   | Пупкин В.В.                 | 123-45-67                        | pupkin@mail.ru           | pupkin | 7c4a8d09ca3762af61e59520943dc26494f8941b |                                    |
| 2   | John Smith                  | 111-22-33                        | smith@gmail.com          | smith  | dd5fef9c1c1da1394d6d34b248c51be2ad740840 |                                    |

РМД поддерживает как декларативное, так и процедурное программирование.

```

CREATE TABLE IF NOT EXISTS `news` (
  `n_uid` int(11) NOT NULL
  AUTO_INCREMENT,
  `n_parent` int(11) NOT NULL,
  `n_dt` int(11) NOT NULL,
  `n_header` text NOT NULL,
  `n_text` text NOT NULL,
  PRIMARY KEY (`n_uid`),
  KEY `n_parent` (`n_parent`),
  KEY `n_dt` (`n_dt`)
) ENGINE=InnoDB DEFAULT CI
AUTO_INCREMENT=100002 ;
  
```

```

DELIMITER //
CREATE TRIGGER `upd_date_ai` AFTER INSERT ON
`news`
FOR EACH ROW BEGIN
    DECLARE old_last_date int;
    SET old_last_date = (SELECT `nr_last_date` from
`news_rubrics` where `nr_uid`=NEW.`n_parent`);
    IF old_last_date < NEW.`n_dt` THEN
        UPDATE `news_rubrics` SET `nr_last_date` =
NEW.`n_dt` where `nr_uid`=NEW.`n_parent`;
    END IF;
END
// 
```

### 1.4.3. Достоинства реляционной модели данных

Простота (единственной информационной конструкцией является отношение).

| su_uid<br>Уникальный<br>идентификатор<br>пользователя | su_name<br>ФИО<br>пользователя | su_phone<br>Телефон<br>пользователя | su_email<br>E-mail<br>пользователя | su_login<br>Логин<br>пользователя | su_password<br>Пароль пользователя       |
|---|--------------------------------|-------------------------------------|------------------------------------|-----------------------------------|--|
| 1   | Пупкин<br>В.В.                 | 123-45-67                           | pupkin@mail.ru                     | pupkin                            | 7c4a8d09ca3762af61e59520943dc26494f8941b |
| 2   | John<br>Smith                  | 111-22-33                           | smith@gmail.com                    | smith                             | dd5fef9c1c1da1394d6d34b248c51be2ad740840 |

При проектировании применяются строгие правила, базирующие на математическом аппарате.

Нормальные  
формы

Теория  
множеств

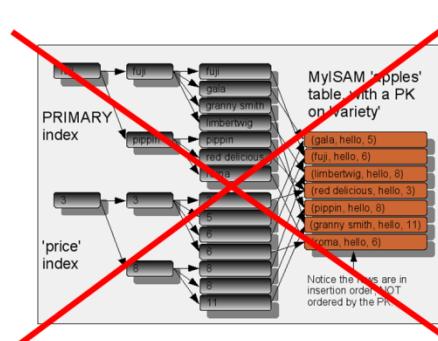
Теория  
зависимостей

Реляционная  
алгебра

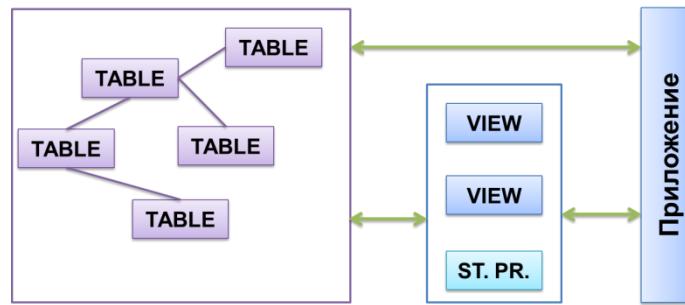
Реляционное  
счисление

РМД обеспечивает полную независимость данных (приложениям «не нужно знать» ничего о внутренних форматах хранения, способах обработки данных и т.п.)

SELECT \* from  
`table\_name`

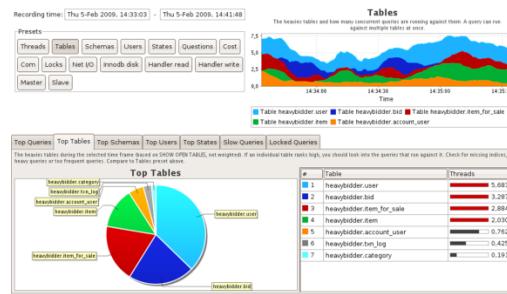


Изменения в структуре БД могут быть скрыты от внешних приложений (в т.ч. с использованием т.н. «представлений» (view) и хранимых подпрограмм (stored routines)).

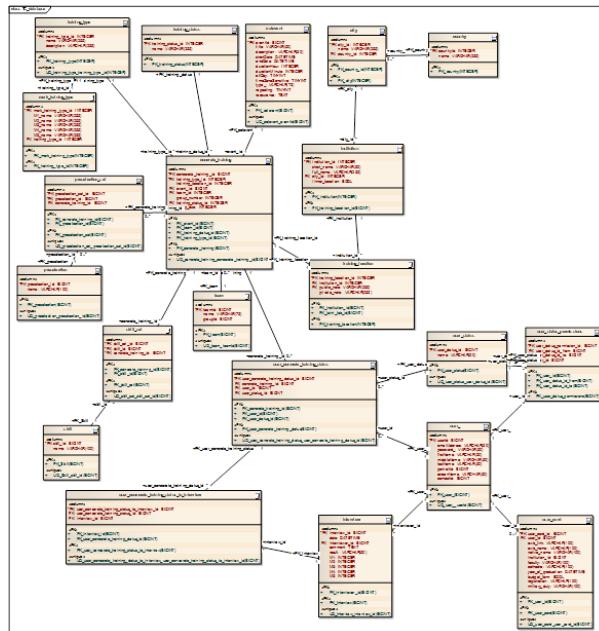


#### 1.4.4. Недостатки реляционной модели данных

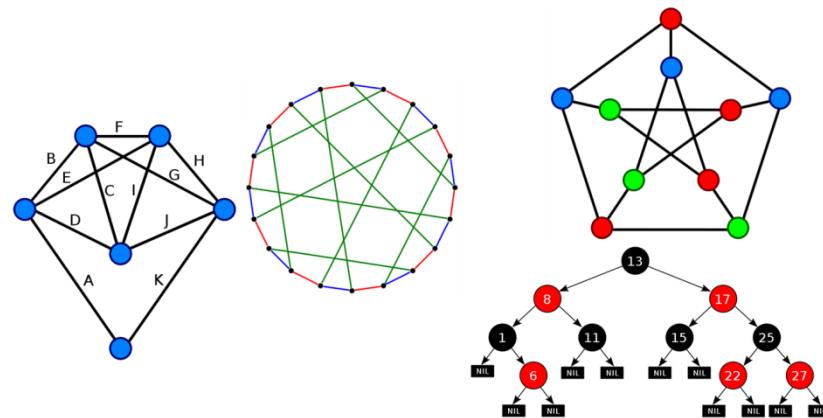
Относительно низкая скорость доступа к данным и использование большого объема внешней памяти.



Трудность понимания структуры данных из-за появления большого количества таблиц в результате логического проектирования.

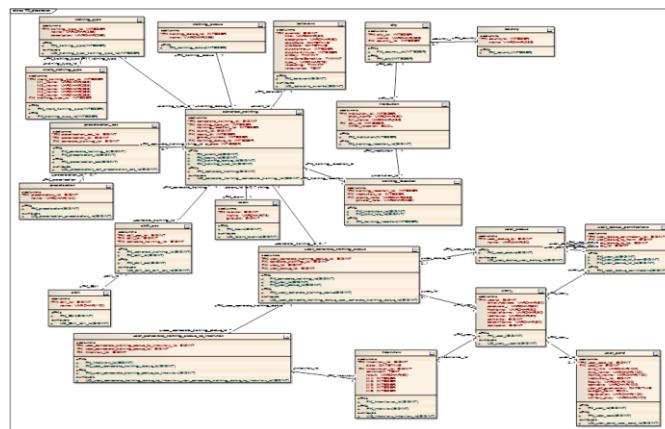


Невозможность или крайне высокая сложность представления в виде таблиц некоторых предметных областей.



#### 1.4.5. Реляционные базы данных

**Реляционные базы данных** (relational databases) – базы данных, основанные на только что рассмотренной реляционной модели данных.



## 1.5. Реляционные системы управления базами данных

Любая реляционная СУБД (это – самый распространённый вид СУБД) должна удовлетворять определённым «12-ти правилам Кодда». На практике некоторые СУБД отступают от тех или иных правил, но лишь по особой необходимости.

### 0. Основное правило (Foundation Rule)

Реляционная СУБД должна управлять данными, используя ТОЛЬКО реляционные механизмы.

```
SELECT * from `table_name`
```

```
file = fopen('data.dat', 'rb+);
```

#### 1. Явное представление данных (The Information Rule)

Данные представлены в таблицах, содержимое ячеек атомарно, порядок строк таблицы не влияет на смысл данных.

| sp_name  | Имя страны | sp_name  | Имя страны | sp_name  | Имя страны |
|----------|------------|----------|------------|----------|------------|
| О фирме  |            | Юрлицам  |            | Test     |            |
| Физлицам |            | Test     |            | Услуги   |            |
| Главная  |            | Физлицам |            | Физлицам |            |
| Юрлицам  |            | О фирме  |            | Главная  |            |
| Test     |            | Услуги   |            | Юрлицам  |            |
| Услуги   |            | Главная  |            | О фирме  |            |

#### 2. Гарантированный доступ к данным (Guaranteed Access Rule)

К каждому элементу данных должен быть гарантирован доступ с помощью комбинации имени БД, таблицы, идентификатора строки и имени столбца.

```
SELECT `field_name` from `db_name`.`table_name`  
where `primary_key`=value
```

#### 3. Обработка неизвестных значений (Treatment of Null Values)

Неизвестные значения NULL должны поддерживаться для всех типов данных при выполнении любых операций.

Например, для числовых данных NULL'ы не должны рассматриваться как нули, а для символьных – как пустые строки.

```
SELECT * FROM `a` RIGHT JOIN `b`  
ON `a`.`name` = `b`.`name`
```

| id   | name | id | name   |
|------|------|----|--------|
| NULL | NULL | 1  | Nathan |
| 1    | John | 2  | John   |
| NULL | NULL | 3  | Bob    |
| 3    | Jack | 4  | Jack   |

### 0. Основное правило (Foundation Rule)

Реляционная СУБД должна управлять данными, используя ТОЛЬКО реляционные механизмы.

#### 1. Явное представление данных (The Information Rule)

Данные представлены в таблицах, содержимое ячеек атомарно, порядок строк таблицы не влияет на смысл данных.

#### 2. Гарантированный доступ к данным (Guaranteed Access Rule)

К каждому элементу данных должен быть гарантирован доступ с помощью комбинации имени БД, таблицы, идентификатора строки и имени столбца.

#### 3. Обработка неизвестных значений (Treatment of Null Values)

Неизвестные значения NULL должны поддерживаться для всех типов данных при выполнении любых операций. Например, для числовых данных NULL'ы не должны рассматриваться как нули, а для символьных – как пустые строки.

#### 4. Доступ к описанию БД в терминах РМД (Active On-Line Catalog Based on the Relational Model)

С помощью стандартных средств производится не только управление самими данными, но и описание их структур (таблиц, связей и т.п.)

```
CREATE TABLE IF NOT EXISTS `news` (
  `n_uid` int(11) NOT NULL AUTO_INCREMENT,
  `n_parent` int(11) NOT NULL,
  `n_dt` int(11) NOT NULL,
  `n_header` text NOT NULL,
  `n_text` text NOT NULL,
  PRIMARY KEY (`n_uid`),
  KEY `n_parent` (`n_parent`),
  KEY `n_dt` (`n_dt`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=100002 ;
```

#### 5. Полнота подмножества языка (Comprehensive Data Sublanguage Rule)

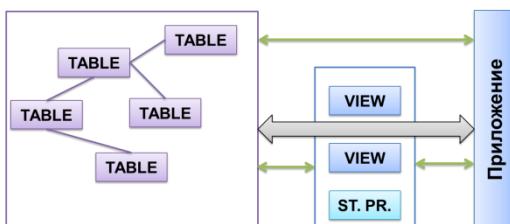
Реляционная СУБД должна поддерживать язык, который:

- имеет линейный синтаксис (может быть записан простым текстом);
- может использоваться как интерактивно, так и в прикладных программах;
- поддерживает все необходимые реляционные операции.



#### 6. Возможность модификации представлений (View Updating Rule)

Представления (view) должны поддерживать все операции манипулирования данными, которые поддерживают реляционные таблицы.



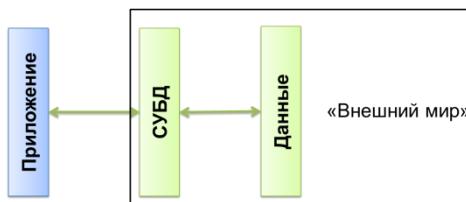
#### 7. Наличие высокоуровневых операций управления данными (High Level Data Manipulation)

Операции вставки, модификации и удаления данных должны поддерживаться не только по отношению к одной строке реляционной таблицы, но по отношению к любому множеству строк.

```
DELETE from `db_name`.`table_name` where
`primary_key`>=999
```

#### 8. Физическая независимость данных (Physical Data Independence)

Приложения не должны зависеть от используемых способов хранения данных на носителях, от аппаратного обеспечения компьютеров, на которых находится реляционная база данных.



## 4. Доступ к описанию БД в терминах РМД (Active On-Line Catalog Based on the Relational Model)

С помощью стандартных средств производится не только управление самими данными, но и описание их структур (таблиц, связей и т.п.)

## 5. Полнота подмножества языка (Comprehensive Data Sublanguage Rule)

Реляционная СУБД должна поддерживать язык, который: имеет линейный синтаксис (может быть записан простым текстом); может использоваться как интерактивно, так и в прикладных программах; поддерживает все необходимые реляционные операции.

## 6. Возможность модификации представлений (View Updating Rule)

Представления (view) должны поддерживать все операции манипулирования данными, которые поддерживают реляционные таблицы.

## 7. Наличие высокоуровневых операций управления данными (High Level Data Manipulation)

Операции вставки, модификации и удаления данных должны поддерживаться не только по отношению к одной строке реляционной таблицы, но по отношению к любому множеству строк.

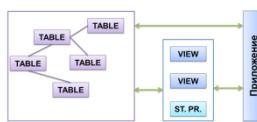
## 8. Физическая независимость данных (Physical Data Independence)

Приложения не должны зависеть от используемых способов хранения данных на носителях, от аппаратного обеспечения компьютеров, на которых находится реляционная база данных.

**9. Логическая независимость данных (Logical Data Independence)**

Представление данных в приложении **не должно зависеть от структуры реляционных таблиц**.

Если в процессе нормализации структура БД меняется, представление (view) **должно обеспечить для приложений доступ к данным так, словно изменений не было.**

**10. Независимость контроля целостности (Integrity Independence)**

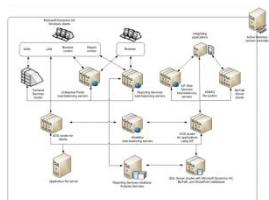
Вся информация, необходимая для поддержания целостности, должна находиться в **словаре данных**.

Язык для работы с данными должен выполнять проверку **входных данных** и автоматически поддерживать целостность данных.

```
ALTER TABLE `news`
ADD CONSTRAINT `news_ibfk_1` FOREIGN KEY (`n_parent`)
REFERENCES `news_rubrics` (`nr_uid`) ON DELETE CASCADE ON
UPDATE CASCADE;
```

**11. Дистрибутивная независимость (Distribution Independence)**

База данных может быть распределённой, может находиться на нескольких компьютерах, переноситься с одного компьютера на другой, и это **не должно оказывать влияния на приложения**.

**12. Согласование языковых уровней (The Non-Subversion Rule)**

Если используется низкоуровневый язык доступа к данным, он **не должен игнорировать правила безопасности и правила целостности**, которые поддерживаются языком более высокого уровня.

```
ALTER TABLE `news`
ADD CONSTRAINT `news_ibfk_1` FOREIGN KEY (`n_parent`)
REFERENCES `news_rubrics` (`nr_uid`) ON DELETE CASCADE ON
UPDATE CASCADE;
```

**9. Логическая независимость данных (Logical Data Independence)**

Представление данных в приложении не должно зависеть от структуры реляционных таблиц. Если в процессе нормализации структура БД меняется, представление (view) должно обеспечить для приложений доступ к данным так, словно изменений не было.

**10. Независимость контроля целостности (Integrity Independence)**

Вся информация, необходимая для поддержания целостности, должна находиться в словаре данных. Язык для работы с данными должен выполнять проверку входных данных и автоматически поддерживать целостность данных.

**11. Дистрибутивная независимость (Distribution Independence)**

База данных может быть распределённой, может находиться на нескольких компьютерах, переноситься с одного компьютера на другой, и это не должно оказывать влияния на приложения.

**12. Согласование языковых уровней (The Non-Subversion Rule)**

Если используется низкоуровневый язык доступа к данным, он не должен игнорировать правила безопасности и правила целостности, которые поддерживаются языком более высокого уровня.

## 1.6. История развития баз данных

### 1-й этап – «большие машины»

Первый этап развития СУБД связан с организацией баз данных на **больших машинах** типа IBM 360/370, ЕС-ЭВМ и т.п.

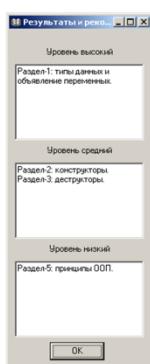
Базы данных хранились во внешней памяти центральной ЭВМ, пользователями этих баз данных были **задачи**, запускаемые в основном в пакетном режиме.



### 2-й этап – «персональные компьютеры»

Эпоха «быдлокодинга» – появилось множество «СУБД-образных» программ, написанных «на скорую руку».

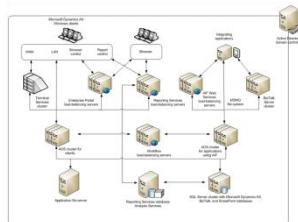
В большинстве своём эти программы были **примитивны**, решали **узкий круг задач**, были ориентированы на однопользовательский режим.



### 3-й этап – «распределённые базы данных»

Остро всталась задача согласованности данных, логически связанных друг с другом, но хранящихся и обрабатывающихся в разных местах.

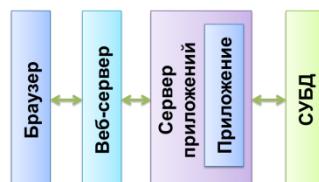
Распределённые СУБД сохранили все преимущества «настольных» СУБД и в то же время позволили организовать параллельную обработку информации и поддержку целостности БД.



### 4-й этап – «многоуровневая архитектура»

Отпадает необходимость использования специализированного клиентского программного обеспечения (**всё берут на себя сервера приложений и сами веб-приложения**).

Клиентской частью для доступа к данным выступают **браузеры и их аналоги**.



### 1-й этап – «большие машины»

Первый этап развития СУБД связан с организацией баз данных на **больших машинах** типа IBM 360/370, ЕС-ЭВМ и т.п.

Базы данных хранились во внешней памяти центральной ЭВМ, пользователями этих баз данных были задачи, запускаемые в основном в пакетном режиме.

### 2-й этап – «персональные компьютеры»

Эпоха «быдлокодинга» – появилось множество «СУБД-образных» программ, написанных «на скорую руку». В большинстве своём эти программы были примитивны, решали узкий круг задач, были ориентированы на однопользовательский режим.

### 3-й этап – «распределённые базы данных»

Остро всталась задача согласованности данных, логически связанных друг с другом, но хранящихся и обрабатывающихся в разных местах. Распределённые СУБД сохранили все преимущества «настольных» СУБД и в то же время позволили организовать параллельную обработку информации и поддержку целостности БД.

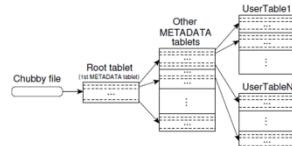
### 4-й этап – «многоуровневая архитектура»

Отпадает необходимость использования специализированного клиентского программного обеспечения (**всё берут на себя сервера приложений и сами веб-приложения**). Клиентской частью для доступа к данным выступают браузеры и их аналоги.

**5-й этап – «NoSQL (not only SQL)»**

Требуется очень быстро и надёжно обрабатывать огромные наборы данных, проводить сложный анализ данных на уровне СУБД и т.д.

Возрождаются иерархические БД, «картонки» и прочие частные решения, оптимизированные под определённый круг задач.

**5-й этап – «NoSQL (not only SQL)»**

Требуется очень быстро и надёжно обрабатывать огромные наборы данных, проводить сложный анализ данных на уровне СУБД и т.д. Возрождаются иерархические БД, «картонки» и прочие частные решения, оптимизированные под определённый круг задач.

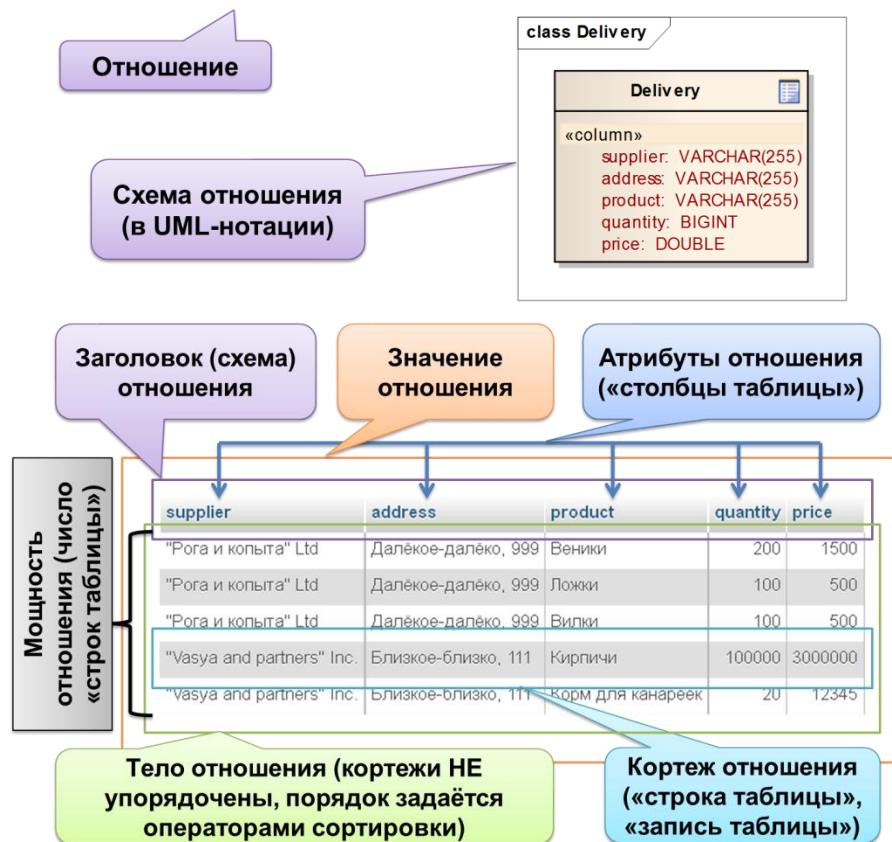
## Тема 2: Отношения, ключи, связи

### 2.1. Отношения

**Отношение, сущность** (relation, entity) R степени n – подмножество декартового произведения множеств D<sub>1</sub>, D<sub>2</sub>, … , D<sub>n</sub> ( $n \geq 1$ ). Исходные множества D<sub>1</sub>, D<sub>2</sub>, … , D<sub>n</sub> называются доменами (в СУБД – «типа данных»).

|                     | D <sub>1</sub>            | D <sub>2</sub>      | D <sub>3</sub>    | D <sub>4</sub> | D <sub>5</sub> |
|---------------------|---------------------------|---------------------|-------------------|----------------|----------------|
|                     | supplier                  | address             | product           | quantity       | price          |
| <b>Наб. зн. 1</b>   | "Рога и копыта" Ltd       | Далёкое-далёко, 999 | Веники            | 200            | 1500           |
| <b>Наб. зн. 2</b>   | "Рога и копыта" Ltd       | Далёкое-далёко, 999 | Ложки             | 100            | 500            |
| <b>Наб. зн. ...</b> | "Рога и копыта" Ltd       | Далёкое-далёко, 999 | Вилки             | 100            | 500            |
|                     | "Vasya and partners" Inc. | Близкое-близко, 111 | Кирпичи           | 100000         | 3000000        |
|                     | "Vasya and partners" Inc. | Близкое-близко, 111 | Корм для канареек | 20             | 12345          |

| supplier                  | address             | product           | quantity | price   |
|---------------------------|---------------------|-------------------|----------|---------|
| "Pora и копыта" Ltd       | Далёкое-далёко, 999 | Веники            | 200      | 1500    |
| "Pora и копыта" Ltd       | Далёкое-далёко, 999 | Ложки             | 100      | 500     |
| "Pora и копыта" Ltd       | Далёкое-далёко, 999 | Вилки             | 100      | 500     |
| "Vasya and partners" Inc. | Близкое-близко, 111 | Кирпичи           | 100000   | 3000000 |
| "Vasya and partners" Inc. | Близкое-близко, 111 | Корм для канареек | 20       | 12345   |



### 2.1.2. Отношения в реляционной теории и в реальности

Теория реляционных БД говорит, что таблица может считаться отношением, если:

В реальности в таблицу БД можно поместить 2+ одинаковые строки. Этого не делают, но всё же...

- В таблице нет двух одинаковых строк.
- У таблицы есть столбцы, соответствующие атрибутам отношения.
- Каждый атрибут-столбец имеет уникальное имя.
- Порядок строк в таблице произвольный.

OK

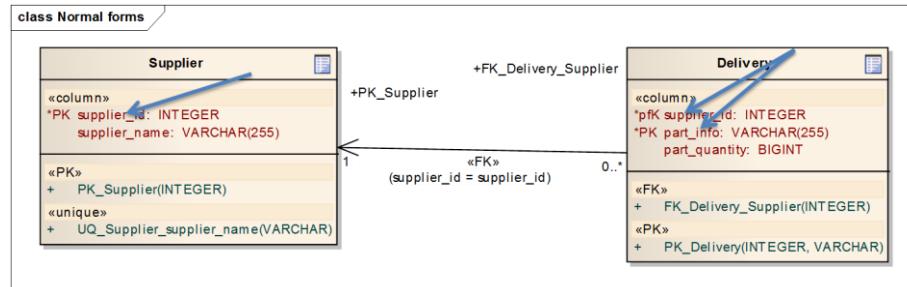
OK

Например, JOIN'ы в MySQL просто возвращают таблицы с 2+ одинаковыми именами.

## 2.2. Ключи

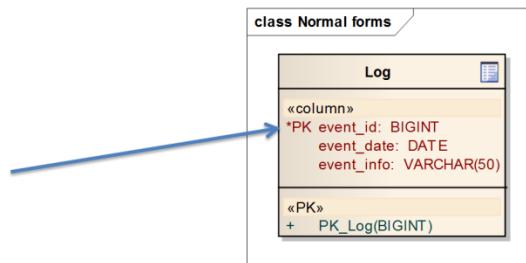
### 2.2.1. Определение

**Ключ** (key) – атрибут (или совокупность атрибутов) отношения, обладающий некоторыми специфическими свойствами, зависящими от вида ключа.



### 2.2.2. Виды ключей

**Первичный ключ** (primary key, PK) – минимальное множество атрибутов, являющееся подмножеством заголовка данного отношения, составное значение которых уникально определяет кортеж отношения.



Значение первичного ключа не может повторяться в двух и более строках.

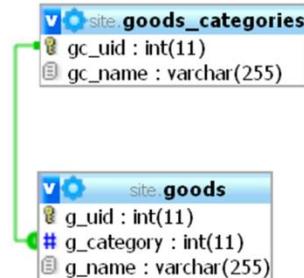
| sp_uid<br>Идентификатор страницы | sp_parent<br>Ссылка на родительскую страницу | sp_name<br>Имя страницы |
|----------------------------------|--|-------------------------|
| 1                                | NULL   | Главная                 |
| 2                                | 1  | О фирме                 |
| 3                                | 1  | Услуги                  |
| 4                                | 3  | Юрлицам                 |
| 5                                | 3  | Физлицам                |
| 6                                | 5  | Test                    |

Первичный ключ служит для:

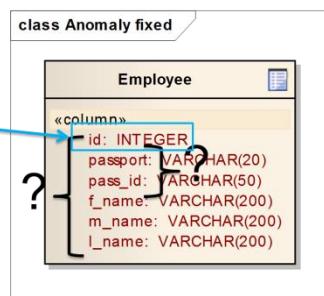
Однозначной идентификации строки таблицы

| sp_uid<br>Идентификатор страницы | sp_parent<br>Ссылка на родительскую страницу | sp_name<br>Имя страницы |
|----------------------------------|--|-------------------------|
| 1                                | NULL   | Главная                 |
| 2                                | 1  | О фирме                 |
| 3                                | 1  | Услуги                  |
| 4                                | 3  | Юрлицам                 |
| 5                                | 3  | Физлицам                |
| 6                                | 5  | Test                    |

Организации связей между таблицами



Из всех вариантов нужно выбирать в качестве первичного ключа самое «короткое» поле или самую «короткую» комбинацию полей.



**Простой первичный ключ** (simple primary key) – первичный ключ, состоящий из единственного поля таблицы (атрибута отношения).

| passport | fml_name     |
|----------|--------------|
| MA334455 | Сидоров С.С. |
| MM991122 | Сидоров С.С. |
| MM998877 | Петров П.П.  |
| MP112233 | Иванов И.И.  |
| MP334455 | Иванов И.И.  |

**Составной первичный ключ** (compound PK, composite PK, concatenated PK) – первичный ключ, состоящий из нескольких полей таблицы.

| passport | country | fml_name     |
|----------|---------|--------------|
| MA998877 | BY      | Сидоров С.С. |
| MM112233 | BY      | Иванов И.И.  |
| MM112233 | RU      | Петров П.П.  |

Последовательность полей в составных PK имеет значение: СУБД может проводить поиск очень быстро целиком по всему составному PK или по первому полю, входящему в его состав.

Т.о. следует определить, по какому полю поиск производится чаще, и так создать ключ:

- по номеру паспорта => {passport, country}
- по стране => {country, passport}

**Естественный первичный ключ** (natural PK) – поле таблицы, хранящее полезные данные.

| passport | fml_name     |
|----------|--------------|
| MA334455 | Сидоров С.С. |
| MM991122 | Сидоров С.С. |
| MM998877 | Петров П.П.  |
| MP112233 | Иванов И.И.  |
| MP334455 | Иванов И.И.  |

У естественных первичных ключей есть **недостатки**:

Большой размер.



Необходимость каскадных изменений.



| passport | fml_name     | passport | money   |
|----------|--------------|----------|---------|
| MA334455 | Сидоров С.С. | MA334455 | 999     |
| MM991122 | Сидоров С.С. | MA334455 | 5754    |
|          |              | MA334455 | 6433634 |
|          |              | MM991122 | 454534  |

Невозможность вставки данных при отсутствии части информации.



**Синтетический** (суррогатный) первичный ключ (*synthetic, surrogate PK*) – искусственно добавленное в таблицу поле, единственная задача которого – быть первичным ключом.

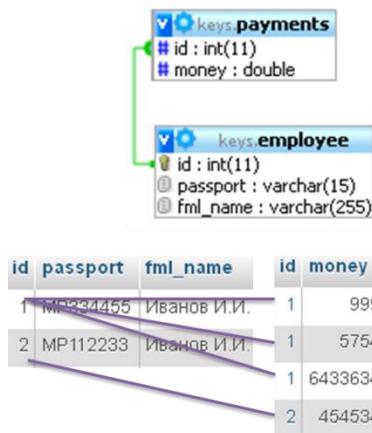
| <b>id</b> | <b>passport</b> | <b>fml_name</b>          |
|-----------|-----------------|--------------------------|
| 1         | MP334455        | Иванов И.И.              |
| 2         | MP112233        | Иванов И.И.              |
| 3         | MM998877        | Петров П.П.              |
| 4         | MM991122        | Сидоров С.С.             |
| 5         | MA334455        | Сидоров С.С.             |
| 6         |                 | Какой-то мужик с бодуна, |

У синтетических ключей есть **преимущества**:

Малый размер.



Нет необходимости каскадных изменений.



Можно вставить данные при отсутствии части информации.

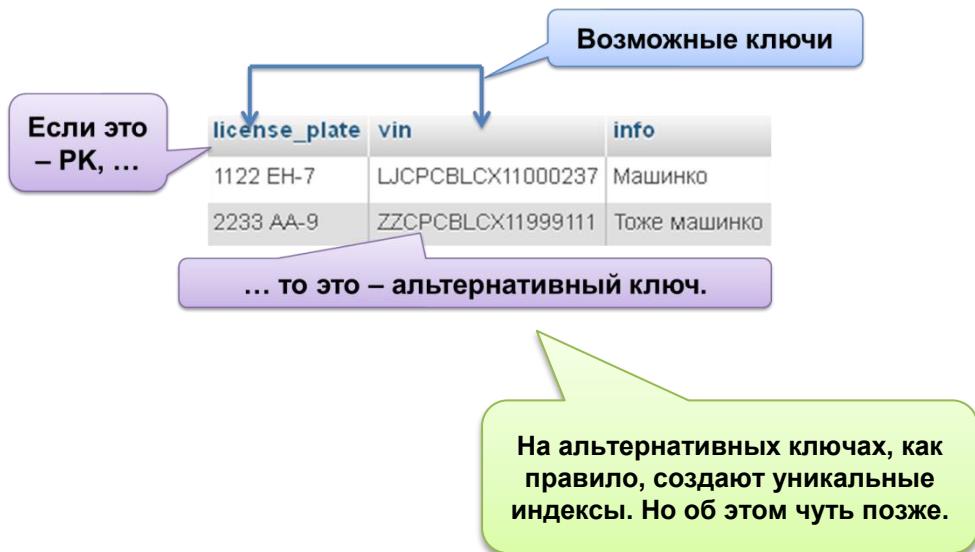
| <b>id</b> | <b>passport</b> | <b>fml_name</b> | <b>id</b> | <b>money</b> |
|-----------|-----------------|-----------------|-----------|--------------|
| 1         | MP334455        | Иванов И.И.     | 1         | 999          |
| 2         | MP112233        | Иванов И.И.     | 1         | 5754         |

| <b>id</b> | <b>passport</b> | <b>fml_name</b>                                   |
|-----------|-----------------|---|
| 6         |                 | Какой-то мужик с бодуна, готов работать двоником, |

**Возможный ключ** (possible key) – поле или совокупность полей с уникальными значениями, кандидат в первичные ключи.

**Альтернативный ключ** (alternate key) – поле или совокупность полей с уникальными значениями, не выбранные в качестве первичного ключа.



**Интеллектуальный первичный ключ** (intelligent primary key) – поле, значение которого формируется на основе значений других полей.

| ipk    | f_name    | m_name        | l_name      |
|--------|-----------|---------------|-------------|
| ИвИвИв | Иван      | Иванович      | Иванов      |
| СиСиСи | Сидор     | Сидорович     | Сидоров     |
| ПёПеПе | Петр      | Петрович      | Петров      |
| АлАлАл | Александр | Александрович | Александров |

С интеллектуальными ключами есть проблемы из-за которых этот вид ключей практически не используется.

Их нужно генерировать и обновлять.



Высока вероятность коллизий (совпадений значений).

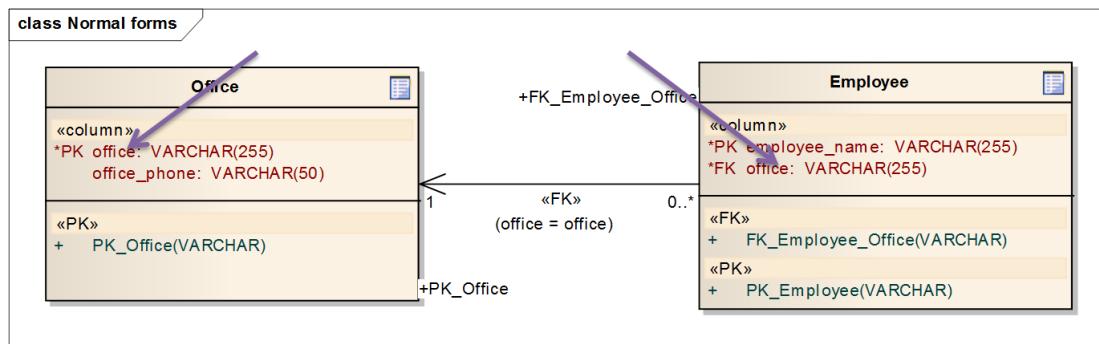
|        |           |               |             |
|--------|-----------|---------------|-------------|
| АлАлАл | Александр | Александрович | Александров |
| АлАлАл | Алексей   | Алексеевич    | Алексеев    |

**Внешний ключ** (foreign key) – поле таблицы, предназначенное для хранения значения первичного ключа другой таблицы с целью организации связи между этими таблицами.



Миграция первичного ключа из родительской таблицы в дочернюю либо производится:

- автоматически (ErWin);
- вручную (Sparx EA) – создать в дочерней таблице поле такого же типа (имя можно брать другое!!!), что и мигрирующий РК (включая регистровую чувствительность, «знаковость» и т.п.), и указать его в качестве внешнего ключа для проведённой между таблицами связи.



**Рекурсивный внешний ключ** (recursive foreign key) – внешний ключ, полученный из поля этой же таблицы.



Классика применения RFK – хранение древовидных структур (например, структуры сайта).



РК

| sp_uid Идентификатор страницы | sp_parent Ссылка на родительскую страницу | sp_name Имя страницы |
|-------------------------------|---|----------------------|
| 1                             | NULL                                      | Главная              |
| 2                             | 1   | О фирме              |
| 3                             | 1   | Услуги               |
| 4                             | 3   | Юрлицам              |
| 5                             | 3   | Физлицам             |
| 6                             | 5   | Test                 |

RFK

## 2.3. Индексы

### 2.3.1. Определение

**Индекс** (index) – объект БД, создаваемый с целью повышения производительности поиска данных. Простая аналогия «из жизни» – карта города. Можно посмотреть по карте, где находится искомый объект, а не бродить в поисках по всему городу.

#### Преимущества индексов

Размер индексов позволяет разместить их в оперативной памяти.

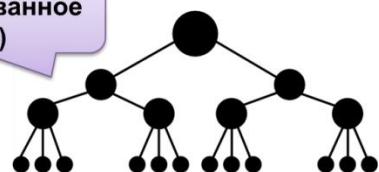
Структура индексов оптимизирована для выполнения операций поиска.

Индексы значительно ускоряют операции поиска данных в БД.

SHOW TABLE STATUS FROM `dbname`

| Rows       | Data    | Index  |
|------------|---------|--------|
| 1+ Million | >100 Mb | <20 Mb |

B-Tree  
(сбалансированное дерево)



Иногда – на 2-3 порядка.  
Но так бывает не всегда,  
нужно исследовать  
ситуацию.

#### Недостатки индексов

Занимают оперативную память.

SHOW TABLE STATUS FROM `dbname`

| Rows       | Data    | Index  |
|------------|---------|--------|
| 1+ Million | >100 Mb | <20 Mb |

В большинстве случаев значительно замедляют операции вставки, обновления, удаления, т.к. требуется обновлять сам индекс.

Иногда – в разы и на порядки.  
С этим можно «бороться»,  
отключая индексы на момент  
вставки большого числа  
записей. В MySQL – так:  
ALTER TABLE `table\_name`  
DISABLE KEYS;  
А потом:  
ALTER TABLE `table\_name`  
ENABLE KEYS;

## Какие индексы создавать?

Признаки того, что индекс нужен:

- операции чтения из таблицы выполняются гораздо чаще, чем операции модификации;
- поле или совокупность полей часто фигурируют в запросах в секции WHERE;
- исследование показало, что наличие индекса повышает производительность.

Проведём исследование. Для таблицы со следующей структурой сначала не будем создавать индексов, кроме первичного ключа:

| Column              | Type         | Collation       |
|---------------------|--------------|-----------------|
| <b>n_uid</b>        | int(11)      |                 |
| <b>n_rubric</b>     | int(11)      |                 |
| <b>n_dt</b>         | int(11)      |                 |
| <b>n_title</b>      | text         | utf8_general_ci |
| <b>n_annotation</b> | text         | utf8_general_ci |
| <b>n_author</b>     | varchar(255) | utf8_general_ci |
| <b>n_text</b>       | text         | utf8_general_ci |
| <b>n_source</b>     | text         | utf8_general_ci |

| Выполним по 1000 раз  | Среднее время, с |
|---|------------------|
| INSERT по 1000 строк  | 0.027289         |
| SELECT * from `news` where `n_rubric`='...'   | 4.035899         |
| SELECT * from `news` where `n_rubric`='...' AND `n_dt`>='...' AND `n_dt`<='...'                     | 4.065648         |
| SELECT * from `news` where `n_dt`>='...' AND `n_dt`<='...'  | 4.508579         |
| SELECT * from `news` where `n_title`='...' AND `n_dt`>='...' AND `n_dt`<='...'                      | 4.207702         |
| SELECT * from `news` where `n_title`='...' AND `n_author`='...' AND `n_dt`>='...' AND `n_dt`<='...' | 4.187432         |
| SELECT * from `news` where `n_title`='...' AND `n_author`='...'                                     | 4.210264         |
| SELECT * from `news` where `n_title`='...'  | 4.173025         |
| SELECT * from `news` where `n_author`='...'   | 4.161251         |

Создадим следующие индексы:

- n\_rubric
- n\_rubric, n\_dt
- n\_title, n\_dt
- n\_dt, n\_author

n\_rubric лишний, т.к. есть {n\_rubric, n\_dt}, и первое поле в индексе может использоваться как «самостоятельно проиндексированное»

| Выполним по 1000 раз  | БЫЛО<br>среднее время,<br>с | СТАЛО<br>среднее время,<br>с |
|---|-----------------------------|------------------------------|
| INSERT по 1000 строк  | 0.027289                    | 0.896445                     |
| SELECT * from `news` where<br>`n_rubric`='...'  | 4.035899                    | 1.200757                     |
| SELECT * from `news` where<br>`n_rubric`='...' AND `n_dt`>='...' AND<br>`n_dt`<='...'                     | 4.065648                    | 0.207999                     |
| SELECT * from `news` where `n_dt`>='...'<br>AND `n_dt`<='...'   | 4.508579                    | 4.318613                     |
| SELECT * from `news` where `n_title`='...'<br>AND `n_dt`>='...' AND `n_dt`<='...'                         | 4.207702                    | 0.003918                     |
| SELECT * from `news` where `n_title`='...'<br>AND `n_author`='...' AND `n_dt`>='...' AND<br>`n_dt`<='...' | 4.18752                     | 0.000468                     |
| SELECT * from `news` where `n_title`='...'<br>AND `n_author`='...'  | 4.210264                    | 0.000909                     |
| SELECT * from `news` where<br>`n_author`='...'  | 4.173025                    | 0.000279                     |
| SELECT * from `news` where<br>`n_author`='...'  | 4.161251                    | 6.567766                     |

Индекс «сработал», просто в выборку попадает очень много записей, извлечение которых отнимает времени.

При поиске по этому полю применимых индексов нет.

Итак, в общем случае индексы лучше создавать, чем не создавать 😊.

**Поля и их комбинации, на которых лучше создать индексы, определяются исходя из наиболее часто выполняемых запросов на чтение.**

Проверить, какие индексы использует СУБД, можно (в MySQL) командой EXPLAIN запрос;

**EXPLAIN SELECT \* from `news` where `n\_author`='...'**

| <b>id</b> | <b>select_type</b> | <b>table</b> | <b>type</b> | <b>possible_keys</b> | <b>key</b> | <b>key_len</b> | <b>ref</b> | <b>rows</b> | <b>Extra</b>        |
|-----------|--------------------|--------------|-------------|----------------------|------------|----------------|------------|-------------|---------------------|
| 1         | SIMPLE             | news         | ALL         |                      | NULL       | NULL           | NULL       | NULL        | 1000000 Using where |

### 2.3.2. Виды индексов

**Уникальные (unique)** – запрещают вставку одинаковых значений в поле таблицы. Как правило, создаются на «альтернативном ПК».

**Неуникальные (non-unique)** – просто индексы ☺, созданные для ускорения поиска.

| wui | wnui |
|-----|------|
| 1   | 2    |
| 3   | 2    |
| 4   | 2    |
| 7   | 3    |
| 9   | 3    |

**Кластерные (cluster index)** – строятся на поле, по значению которого упорядочена таблица. В таблице может быть только один кластерный индекс (и это, как правило – ПК).

**Некластерные (non-cluster index)** – строятся на произвольном неупорядоченном поле таблицы.

**Простые (simple index)** – строятся на одном поле.

**Составные, сложные (complex index)** – строятся на нескольких полях или даже на выражениях.

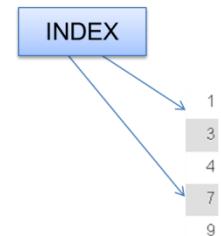
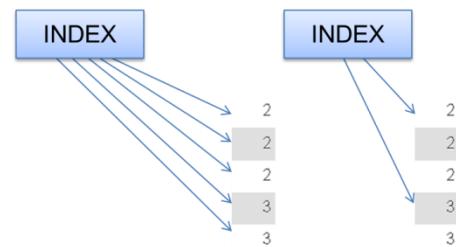
| sn_uid | sn_dt      | sn_title        |
|--------|------------|-----------------|
| 1      | 118758071  | Title 64379878  |
| 2      | 121669144  | Title 64046656  |
| 3      | 1569875561 | Title 344103318 |
| 4      | 1631586361 | Title 647355496 |
| 5      | 388175780  | Title 8939791   |
| 6      | 160759710  | Title 288420257 |

| sn_uid | sn_dt      | sn_title        |
|--------|------------|-----------------|
| 1      | 118758071  | Title 64379878  |
| 2      | 121669144  | Title 64046656  |
| 3      | 1569875561 | Title 344103318 |
| 4      | 1631586361 | Title 647355496 |
| 5      | 388175780  | Title 8939791   |
| 6      | 160759710  | Title 288420257 |

Сложный индекс по  
UPPER(`sn\_title`)  
**(HE поддерживается MySQL)**

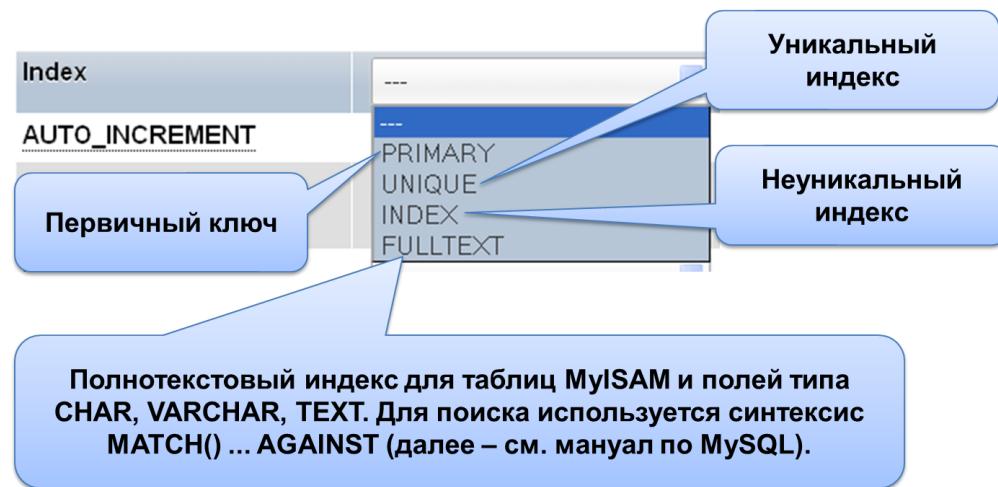
**Плотные** (dense index) – указывают на конкретную запись в таблице или блок записей с одинаковыми значениями индексированного поля.

**Неплотные**, редкие (sparse index) – указывают на блок (отсортированных) записей.



### 2.3.3. Управление индексами в MySQL

MySQL на текущий момент поддерживает четыре типа индексов:

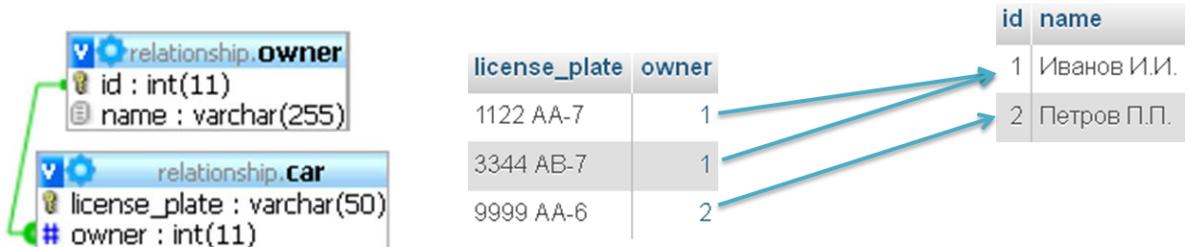


## 2.4. Связи

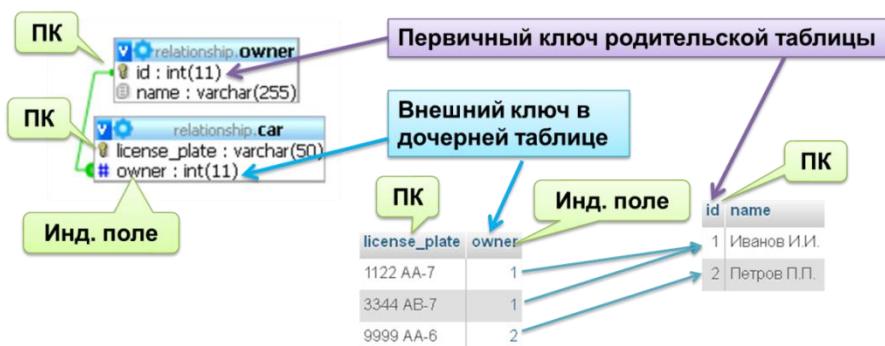
### 2.4.1. Определение

Связь (relationship) – ассоциация, установленная между двумя и более сущностями (relations, entities).

Простая аналогия «из жизни» – указание в описании автомобиля информации о его владельце.



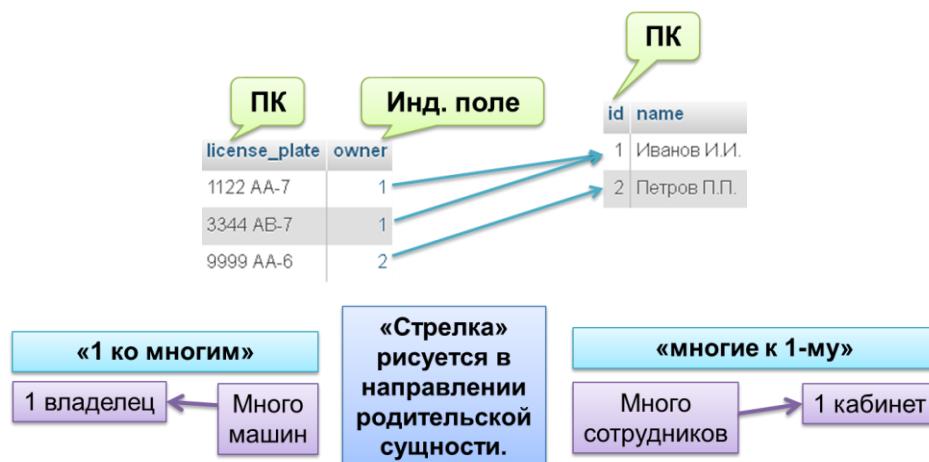
Связь организуется за счёт миграции первичного ключа родительской таблицы в дочернюю таблицу. Получившееся в результате поле называется внешним ключом (foreign key).



### 2.4.2. Виды связей

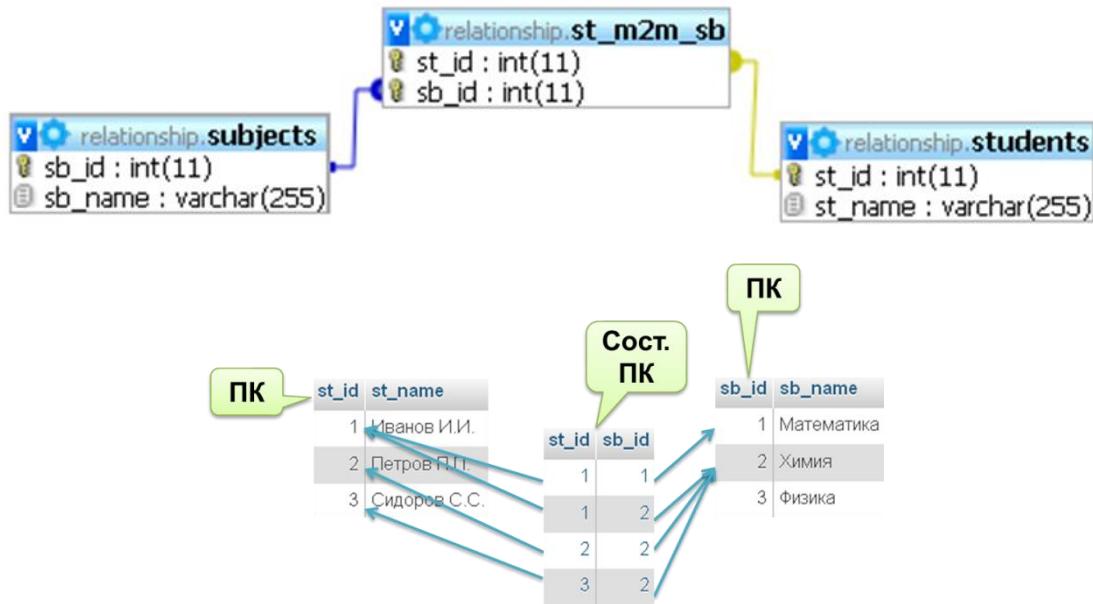
**Связь «1 ко многим»** («многие к 1-му») определяет ситуацию, когда одной записи родительской таблицы соответствует несколько записей дочерней таблицы.

Разница между «1 ко многим» и «многие к 1-му» – исключительно в том, «кто является главным».



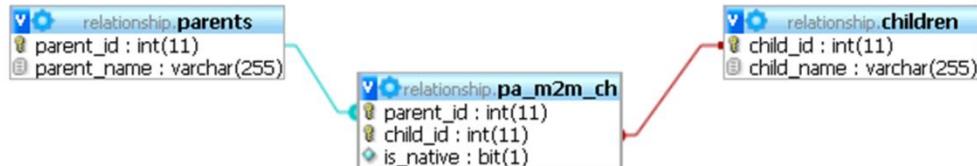
Связь «многие ко многим» определяет ситуацию, когда любой записи одной таблицы может соответствовать много записей другой таблицы и наоборот.

Связь «М-М» – исключительно «человеческое» понятие и в реальных БД реализуется через две связи «1-М».



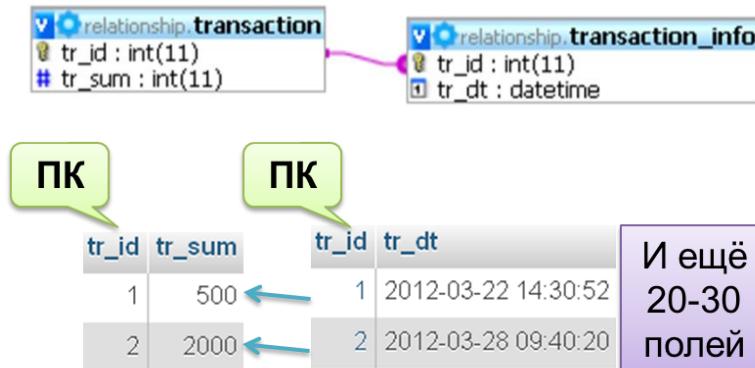
Связь «М-М» может также отражать некие свои свойства, не характерные для связываемых сущностей.

Например, в связи «родители-дети» можно отразить признак родства, который является именно свойством связи, но не родителя или ребёнка.



**Связь «1 к 1-му»** определяет ситуацию, когда любой записи одной таблицы может соответствовать ровно одна запись другой таблицы и наоборот.

Наличие неаргументированной связи «1-1» в простых БД – признак ошибки в формировании структуры.

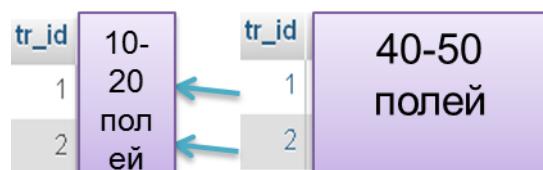


Связь «1-1» оправдана в том случае, если:

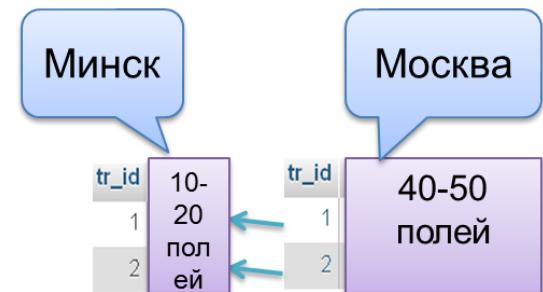
Нужно разнести очень часто и очень редко обрабатываемые данные по разным таблицам для ускорения работы.



Получилась «мега-таблица», количество полей в которой «упирается» в ограничения СУБД.

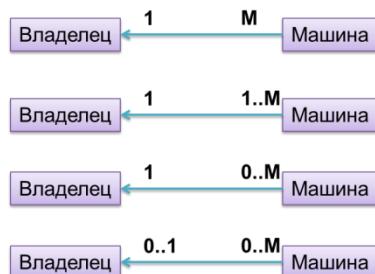


Нужно разнести данные по удалённым друг от друга местам, где они активно обрабатываются.



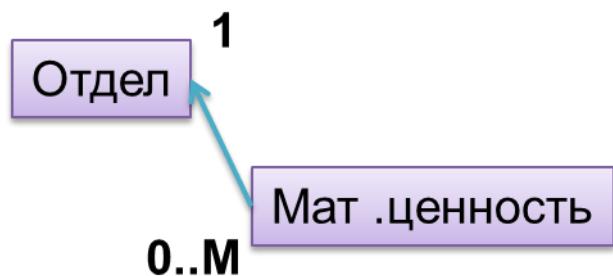
Мощность (кардинальность) связи (relationship cardinality) – указание возможного числа записей в таблице с каждой стороны связи.

При серьёзном проектировании указывают обе (нижнюю и верхнюю) граници с каждой стороны связи.

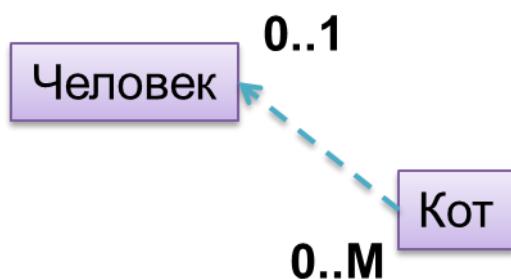


### 2.4.3. Идентифицирующая и неидентифицирующая связи

**Идентифицирующая связь** (identifying relationship) определяет ситуацию, когда запись в дочерней таблице обязана быть связана с записью в родительской таблице.



**Неидентифицирующая связь** (non-identifying relationship) определяет ситуацию, когда запись в дочерней таблице может быть НЕ связана с записью в родительской таблице.



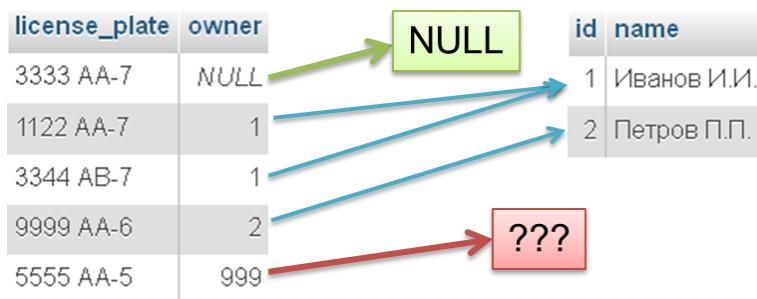
## 2.5. Ссылочная целостность данных

### 2.5.1. Определение

**Ссылочная целостность** (referential integrity) – необходимое качество реляционной БД, заключающееся в отсутствии в любом её отношении внешних ключей, ссылающихся на несуществующие кортежи.

Простым языком: «если ключ на что-то ссылается, это что-то должно существовать».

При наличии ЯВНО прописанных между таблицами связей СУБД не допустит такой ситуации:



### 2.5.2. Каскадные операции

**Каскадные операции** (cascade operations) – специальные ограничения БД, описывающие её поведение в случае удаления записи из родительской таблицы или изменения её первичного ключа.

| Каскадное удаление                  | Каскадное обновление                                   | Установка пустых ключей  | Установка значения по умолчанию  | Запрет каскадной операции  |
|-------------------------------------|--|--|--|--|
| Записи в дочерней таблице удаляются | Значения внешних ключей в дочерней таблице обновляются | Во внешние ключи в дочерней таблице выставляется значение NULL | Во внешние ключи в дочерней таблице выставляется значение по умолчанию | Запись из родительской таблицы нельзя удалить, пока ей соответствуют записи в дочерней таблице |

## Каскадное удаление

При каскадном удалении записи дочерней таблицы, соответствующие удаляемой записи родительской таблицы, тоже удаляются.

## Каскадное обновление

При обновлении первичного ключа родительской таблицы внешние ключи соответствующих записей в дочерней таблице принимают это же новое значение.

## Установка пустых ключей

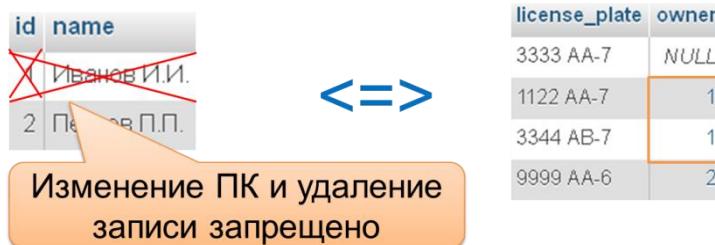
При удалении записи из родительской таблицы во внешние ключи соответствующих записей дочерней таблицы устанавливается значение NULL.

## Установка значения по умолчанию

При удалении записи из родительской таблицы во внешние ключи соответствующих записей дочерней таблицы устанавливается значение по умолчанию.

### Запрет каскадной операции

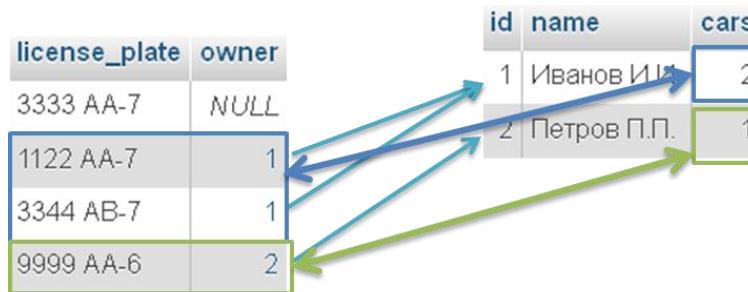
Пока существует хотя бы одна запись в дочерней таблице, соответствующая некоей записи в родительской таблице, эту запись из родительской таблицы нельзя удалить и/или нельзя изменить её первичный ключ.



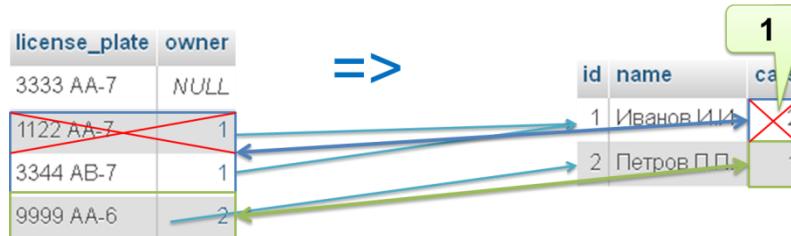
### 2.5.3. Консистентность данных

**Консистентность данных** (data consistency, data validity) – согласованность данных друг с другом: ссылочная целостность и внутренняя непротиворечивость.

Простым языком: фрагменты данных в БД не должны противоречить друг другу.



Консистентность данных, как правило, обеспечивается созданием триггеров, контролирующих операции с таблицами и корректирующих соответствующие данные или блокирующих операцию.



## 2.6. Триггеры

**Триггер** (trigger) – элементарная программа, написанная на некотором расширении языка SQL, и используемая для обеспечения консистентности данных.

Более подробно о триггерах – в разделах, посвящённых языку SQL.

## Виды триггеров

|       | Вставка       | Обновление    | Удаление      |
|-------|---------------|---------------|---------------|
| Перед | BEFORE INSERT | BEFORE UPDATE | BEFORE DELETE |
| После | AFTER INSERT  | AFTER UPDATE  | AFTER DELETE  |

## Задачи триггеров

С использованием триггеров, как правило, решаются задачи, которые нельзя решить только с использованием ссылок, например:

- Обновление агрегированных данных («сколько у владельца машин»).
  - Сложный запрет удаления («в системе должен быть хотя бы один администратор»).
  - Контроль числовых значений («рост – от 50 до 300 см»).
  - ...

## Тема 3: Нормальные формы

### 3.1. Аномалии операций с БД

#### 3.1.1. Важно помнить

Для полноценного понимания материала следует помнить, что схема отношения не эквивалентна самому отношению! Это важно знать и помнить потому, что одной схеме может соответствовать множество отношений.

| Схема   | Отношение   |                   |          |         |          |       |                     |                     |        |     |      |                     |                     |       |     |     |                     |                     |       |     |     |                           |                     |         |        |         |                           |                     |                   |    |       |
|---|---|-------------------|----------|---------|----------|-------|---------------------|---------------------|--------|-----|------|---------------------|---------------------|-------|-----|-----|---------------------|---------------------|-------|-----|-----|---------------------------|---------------------|---------|--------|---------|---------------------------|---------------------|-------------------|----|-------|
| <pre> class Delivery   Delivery   «column»     supplier: VARCHAR(255)     address: VARCHAR(255)     product: VARCHAR(255)     quantity: BIGINT     price: DOUBLE   </pre> | <table border="1"> <thead> <tr> <th>supplier</th><th>address</th><th>product</th><th>quantity</th><th>price</th></tr> </thead> <tbody> <tr> <td>"Рога и копыта" Ltd</td><td>Далёкое-далёко, 999</td><td>Веники</td><td>200</td><td>1500</td></tr> <tr> <td>"Рога и копыта" Ltd</td><td>Далёкое-далёко, 999</td><td>Ложки</td><td>100</td><td>500</td></tr> <tr> <td>"Рога и копыта" Ltd</td><td>Далёкое-далёко, 999</td><td>Вилки</td><td>100</td><td>500</td></tr> <tr> <td>"Vasya and partners" Inc.</td><td>Близкое-близко, 111</td><td>Кирпичи</td><td>100000</td><td>3000000</td></tr> <tr> <td>"Vasya and partners" Inc.</td><td>Близкое-близко, 111</td><td>Корм для канареек</td><td>20</td><td>12345</td></tr> </tbody> </table> | supplier          | address  | product | quantity | price | "Рога и копыта" Ltd | Далёкое-далёко, 999 | Веники | 200 | 1500 | "Рога и копыта" Ltd | Далёкое-далёко, 999 | Ложки | 100 | 500 | "Рога и копыта" Ltd | Далёкое-далёко, 999 | Вилки | 100 | 500 | "Vasya and partners" Inc. | Близкое-близко, 111 | Кирпичи | 100000 | 3000000 | "Vasya and partners" Inc. | Близкое-близко, 111 | Корм для канареек | 20 | 12345 |
| supplier  | address   | product           | quantity | price   |          |       |                     |                     |        |     |      |                     |                     |       |     |     |                     |                     |       |     |     |                           |                     |         |        |         |                           |                     |                   |    |       |
| "Рога и копыта" Ltd   | Далёкое-далёко, 999   | Веники            | 200      | 1500    |          |       |                     |                     |        |     |      |                     |                     |       |     |     |                     |                     |       |     |     |                           |                     |         |        |         |                           |                     |                   |    |       |
| "Рога и копыта" Ltd   | Далёкое-далёко, 999   | Ложки             | 100      | 500     |          |       |                     |                     |        |     |      |                     |                     |       |     |     |                     |                     |       |     |     |                           |                     |         |        |         |                           |                     |                   |    |       |
| "Рога и копыта" Ltd   | Далёкое-далёко, 999   | Вилки             | 100      | 500     |          |       |                     |                     |        |     |      |                     |                     |       |     |     |                     |                     |       |     |     |                           |                     |         |        |         |                           |                     |                   |    |       |
| "Vasya and partners" Inc.   | Близкое-близко, 111   | Кирпичи           | 100000   | 3000000 |          |       |                     |                     |        |     |      |                     |                     |       |     |     |                     |                     |       |     |     |                           |                     |         |        |         |                           |                     |                   |    |       |
| "Vasya and partners" Inc.   | Близкое-близко, 111   | Корм для канареек | 20       | 12345   |          |       |                     |                     |        |     |      |                     |                     |       |     |     |                     |                     |       |     |     |                           |                     |         |        |         |                           |                     |                   |    |       |

#### 3.1.2. Аномалии и их опасность

Допустим, у нас есть отношение, описанное такой схемой:

| class Delivery   |
|--|
| <pre> Delivery «column»   supplier: VARCHAR(255)   address: VARCHAR(255)   product: VARCHAR(255)   quantity: BIGINT   price: DOUBLE   </pre> |

Создадим таблицу и заполним её данными:

| Column   | Type         | Collation       | Attributes | Null | Default |
|----------|--------------|-----------------|------------|------|---------|
| supplier | varchar(255) | utf8_general_ci |            | Yes  | NULL    |
| address  | varchar(255) | utf8_general_ci |            | Yes  | NULL    |
| product  | varchar(255) | utf8_general_ci |            | Yes  | NULL    |
| quantity | bigint(20)   |                 |            | Yes  | NULL    |
| price    | double       |                 |            | Yes  | NULL    |

| supplier                  | address             | product           | quantity | price   |
|---------------------------|---------------------|-------------------|----------|---------|
| "Рога и копыта" Ltd       | Далёкое-далёко, 999 | Веники            | 200      | 1500    |
| "Рога и копыта" Ltd       | Далёкое-далёко, 999 | Ложки             | 100      | 500     |
| "Рога и копыта" Ltd       | Далёкое-далёко, 999 | Вилки             | 100      | 500     |
| "Vasya and partners" Inc. | Близкое-близко, 111 | Кирпичи           | 100000   | 3000000 |
| "Vasya and partners" Inc. | Близкое-близко, 111 | Корм для канареек | 20       | 12345   |

Всё ли тут хорошо? Увы, далеко не всё. Итак...

**Аномалия** (anomaly) – противоречие между моделью предметной области и моделью данных, поддерживаемой средствами конкретной СУБД.

**Аномалия вставки** – при добавлении данных, часть которых у нас отсутствует, мы вынуждены или не выполнять добавление или подставлять пустые или фиктивные данные.

| supplier          | address     | product | quantity | price |
|-------------------|-------------|---------|----------|-------|
| Новый поставщик 1 | Где-то, 123 | NULL    | NULL     | NULL  |
| Новый поставщик 2 | Там-то, 567 |         | 0        | 0     |

**Аномалия обновления** – при обновлении данных мы вынуждены обновлять много строк и рискуем часть строк «забыть обновить».

```
UPDATE `normalisation`.`delivery` SET
`supplier` = '"Рога и копыта" Inc' WHERE
`delivery` . `supplier` = '"Рога и копыта" Ltd'
```

17283545 row(s) affected.

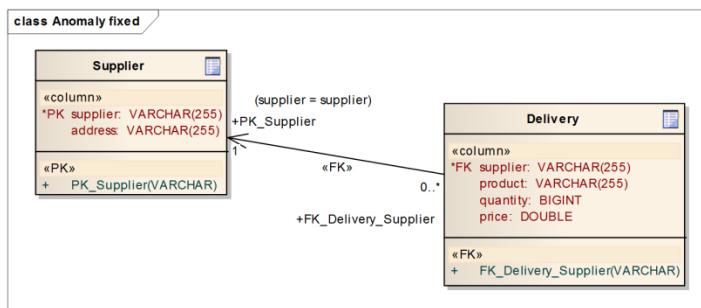
А если где-то «Ltd» было написано с точкой, т.е. «Ltd.»?

**Аномалия удаления** – при удалении части данных мы теряем другую часть, которую не надо было удалять.

Например, сотрудничество с «Рога и копыта Ltd» прекращено, информацию о поставщике нужно удалить, и мы теряем данные о поставках:

| supplier                  | address             | product           | quantity | price   |
|---------------------------|---------------------|-------------------|----------|---------|
| "Рога и копыта" Ltd       | Далёкое-далёко, 999 | Веники            | 200      | 1500    |
| "Рога и копыта" Ltd       | Далёкое-далёко, 999 | Ложки             | 100      | 500     |
| "Рога и копыта" Ltd       | Далёкое-далёко, 999 | Вилки             | 100      | 500     |
| "Vasya and partners" Inc. | Близкое-близко, 111 | Кирпичи           | 100000   | 3000000 |
| "Vasya and partners" Inc. | Близкое-близко, 111 | Корм для канареек | 20       | 12345   |

Изменив схему БД, мы получим следующую картину:



Изменив код БД, мы получим:



| Column          | Type         | Collation       | Attributes | Null | Default |
|-----------------|--------------|-----------------|------------|------|---------|
| <b>supplier</b> | varchar(255) | utf8_general_ci |            | No   | None    |
| <b>address</b>  | varchar(255) | utf8_general_ci |            | Yes  | NULL    |

| Column          | Type         | Collation       | Attributes | Null | Default |
|-----------------|--------------|-----------------|------------|------|---------|
| <b>supplier</b> | varchar(255) | utf8_general_ci |            | No   | None    |
| <b>product</b>  | varchar(255) | utf8_general_ci |            | Yes  | NULL    |
| <b>quantity</b> | bigint(20)   |                 |            | Yes  | NULL    |
| <b>price</b>    | double       |                 |            | Yes  | NULL    |

Чтобы избежать нарушения целостности, необходимо установить каскадное обновление.

Теперь мы можем:

- Добавлять поставщиков, которые ещё не выполнили поставок.
- Изменять информацию о поставщике.
- Удалять поставки и/или поставщиков независимо.

С удалением поставок проблем не будет, а при удалении поставщика нужно установить каскадную операцию SET NULL / RESTRICT в зависимости от желаемого поведения.

Нормализация, о которой скоро пойдёт речь, призвана в том числе устранить подобные аномалии.

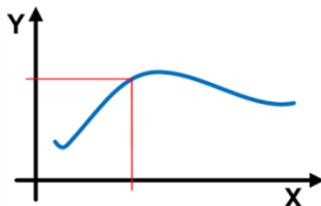
А базируется нормализация на теории зависимостей, которую мы сейчас кратко и рассмотрим.

## 3.2. Теория зависимостей

### 3.2.1. Функциональная зависимость

Если даны два атрибута  $X$  и  $Y$  некоторого отношения, то  $Y$  функционально зависит от  $X$ , если в любой момент времени каждому значению  $X$  соответствует ровно одно значение  $Y$ . Обозначается  $X \rightarrow Y$ .

Примеры:



НомерПаспорта  $\rightarrow$  Фамилия  
ID\_Сотрудника  $\rightarrow$  СтажРаботыВФирме

### 3.2.2. Избыточная функциональная зависимость

Зависимость, заключающая в себе такую информацию, которая может быть получена на основе других зависимостей.

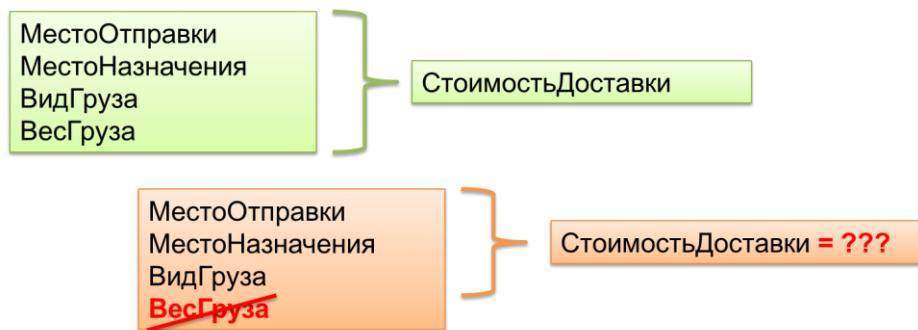
Пример:



### 3.2.3. Полная функциональная зависимость

Функциональная зависимость  $X \rightarrow Y$  является ПОЛНОЙ, если  $Y$  не зависит функционально от любого подмножества  $X$ .

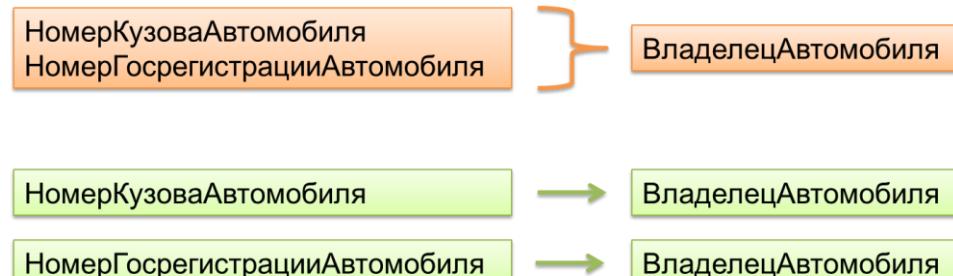
Пример:



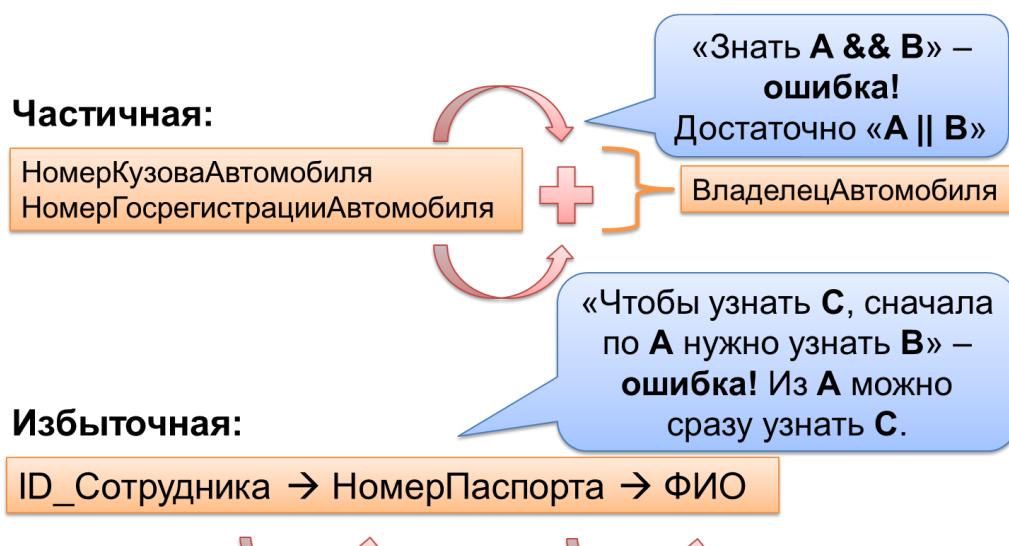
### 3.2.4. Частичная функциональная зависимость

Функциональная зависимость  $X \rightarrow Y$  является ЧАСТИЧНОЙ, если  $Y$  зависит функционально от некоторого подмножества  $X$ .

Пример:



В чём разница между частичной и избыточной зависимостью?



Представьте «тупого сыщика»:

- Частичная: «мы знаем номер госрегистрации машины, теперь если совпадёт номер кузова, то её владелец – Пупкин».
- Избыточная: «мы знаем ID сотрудника, сейчас по нём мы узнаем номер паспорта, а уже по номеру паспорта – фамилию сотрудника».

### 3.2.5. Транзитивная функциональная зависимость

Функциональная зависимость  $X \rightarrow Y$  является транзитивной, если существуют зависимости  $X \rightarrow Z$  и  $Z \rightarrow Y$ , но отсутствует прямая зависимость  $X \rightarrow Y$ .

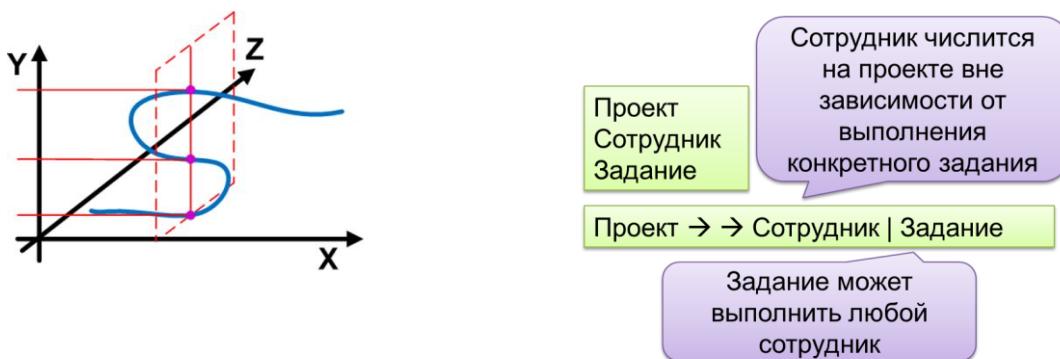
Пример:



### 3.2.6. Многозначная зависимость

Многозначная зависимость  $X \rightarrow\rightarrow Y | Z$  существует в том и только в том случае, если множество значений  $Y$ , соответствующее паре значений  $X$  и  $Z$ , зависит только от  $X$  и не зависит от  $Z$  (то есть для каждого значения атрибута  $X$  существует множество соответствующих значений атрибута  $Y$ ).

Примеры:



Попробуем проще: из того факта, что в отношении существуют кортежи  $R1(X, Y, Z1)$  и  $R2(X, Y1, Z)$ , следует наличие кортежа  $R(X, Y, Z)$ .

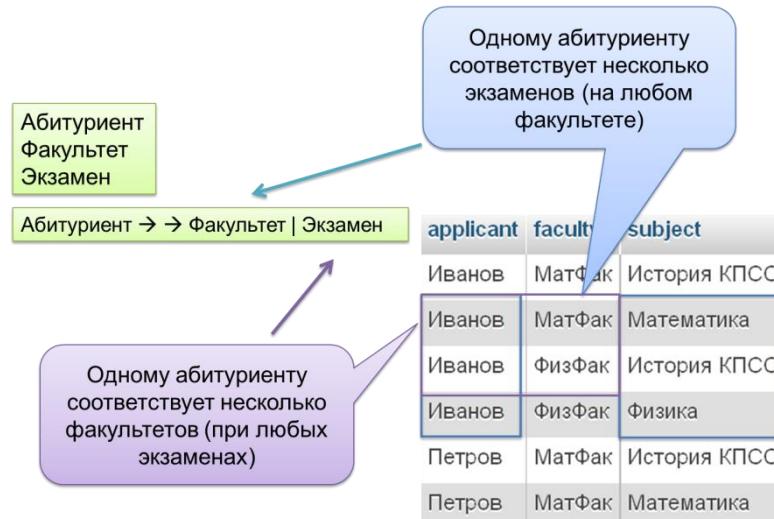
Чтобы это условие выполнялось, необходимо:

- Абитуриент поступает на 2 и более факультета.
- У 2 и более факультетов есть общий экзамен.

Допустим, в одной таблице решили сохранить информацию об абитуриентах, факультетах и сдаваемых экзаменах.

| X      | apply | Y      | faculty | subject      | Z  |
|--------|-------|--------|---------|--------------|----|
| Иванов |       | МатФак |         | История КПСС |    |
| Иванов |       | МатФак |         | Математика   | Z1 |
| Иван   |       | ФизФак |         | История КПСС |    |
| Иванов |       | ФизФак |         | Физика       |    |
| Петров |       | МатФак |         | История КПСС |    |
| Петров |       | МатФак |         | Математика   |    |

Попробуем ещё проще:



### 3.2.7. Тривиальная и нетривиальная многозначная зависимость

#### Тривиальная

Содержит хотя бы одну функциональную зависимость.



#### Нетривиальная

Не содержит функциональных зависимостей.

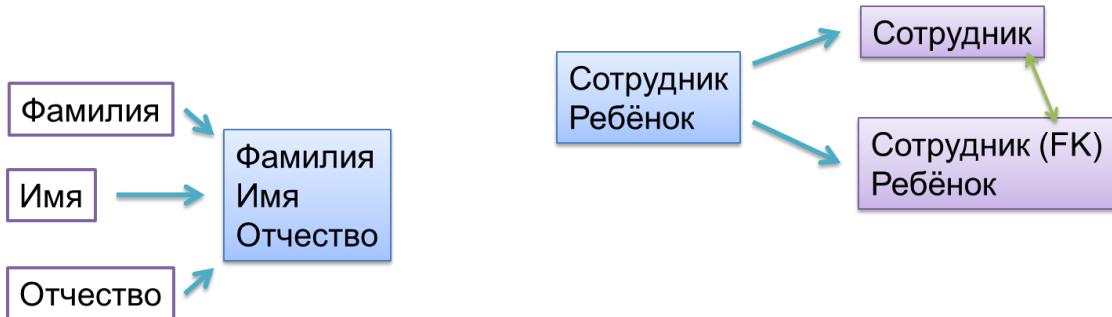


Данные условия «предмет читает один лектор», «книги по предмету не зависят от лектора», «абитуриент может поступать на разные факультеты» и т.п. – это ОГРАНИЧЕНИЯ ПРЕДМЕТНОЙ ОБЛАСТИ, т.е. «так решено». И может быть «решено иначе» в других случаях.

### 3.3. Нормализация и нормальные формы

#### 3.3.1. Определение

**Нормализация** (normalization) – группировка и/или распределение атрибутов по отношениям с целью устранения аномалий операций с БД, обеспечения целостности данных и оптимизации модели БД.



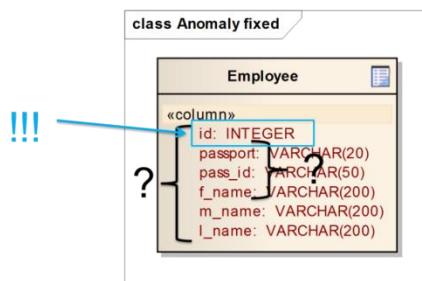
#### 3.3.2. Требования нормализации

Как правило, не существует «единственно правильного способа нормализации» для достаточно сложной БД – у всех решений есть плюсы и минусы. Но желательно придерживаться следующих требований.

Эти требования могут противоречить друг другу, так что не стремитесь выполнить их все любой ценой. Выбирайте то, что важно для вашей конкретной БД!

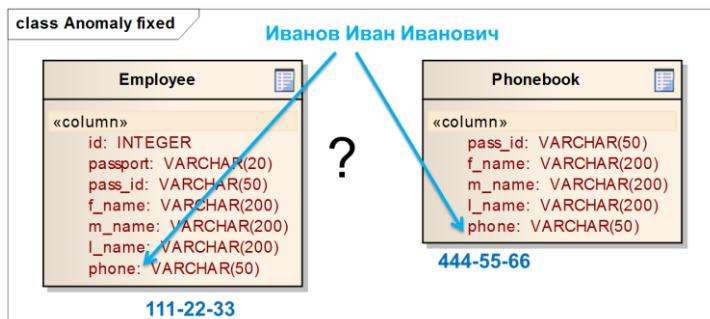
#### Требование минимальности первичных ключей

Первичные ключи отношений должны быть минимальными. Это требование идеально выполняется с введением суррогатных ПК.



#### Требование надёжности данных

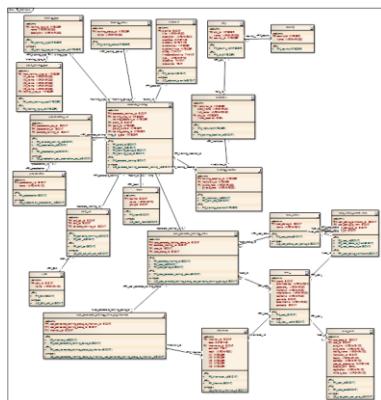
Модель БД должна по возможности минимизировать или устранивать избыточность данных.



## Требование производительности системы

Модель БД должна позволять обеспечивать необходимую производительность операций.

Хммм... Возможно, такая схема, и обеспечивает необходимую производительность, но... есть сомнения.

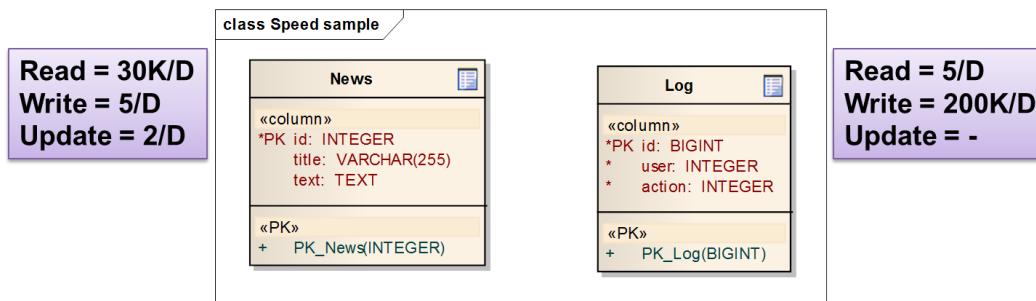


```
SELECT `news_keywords`.`nk_url`,  
`news_keywords`.`nk_name`,  
count(*) as `q` from  
`news_keywords` RIGHT JOIN  
`n_m2m_nk` ON  
`news_keywords`.`nk_uid` = `n_m2m_`  
`n_m2m_nk`.`nk_uid` group by  
`n_m2m_nk`.`nk_uid` order by `q`  
desc, `news_keywords`.`nk_name`  
asc limit 100
```

## Требование сохранения производительности

Разброс времени реакции на различные операции с данными должен быть минимальным.

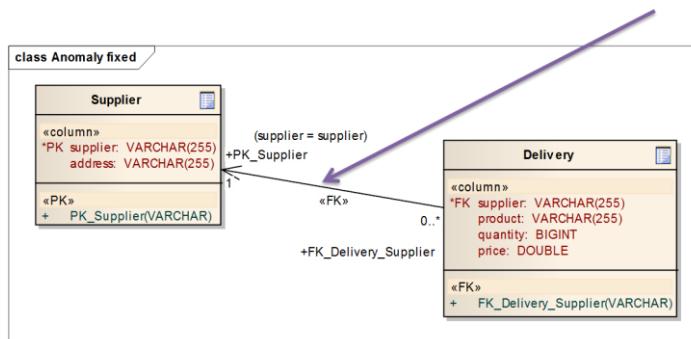
Это требование выполняют крайне редко, т.к. очень часто наблюдается явный «перевес» в сторону каких-то операций при реальном использовании БД.



## Требование непротиворечивости данных

Модель БД должна минимизировать вероятность возникновения противоречивости данных при любых операциях с данными.

Иными словами – связи должны быть установлены ЯВНО!



## Требование гибкости структуры

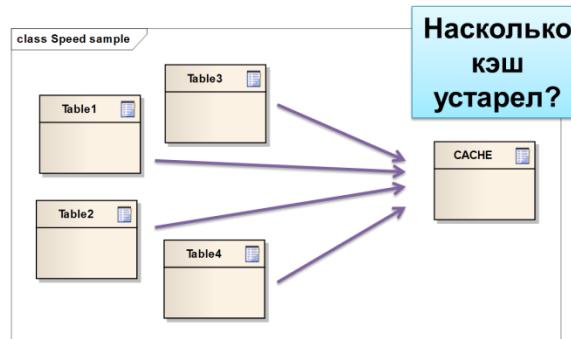
Модель БД должна быть способной к адаптации в случае необходимости внесения изменений.

Это достигается за счёт:

|                                |  |
|--------------------------------|--|
| Мнемоничных имён.              | «abc» – плохо, «user_id» – хорошо  |
| Комментариев.                  | К полям, таблицам, связям, в хранимых подпрограммах и т.д.   |
| Документации.                  | Сначала её не пишут, а потом проклинают тех, кто в своё время не написал ☺.                              |
| Схемы в общепринятой нотации.  | Хорошо подходит UML или IDEF0.   |
| Отсутствия глупых ограничений. | ... вроде ID пользователя размером в 1 байт. И при этом ещё нет документации ☺ – вообще жуть получается. |

## Требование актуальности данных

В каждый момент времени БД должна содержать актуальный набор данных.



### 3.3.3. Нормальные формы низких порядков

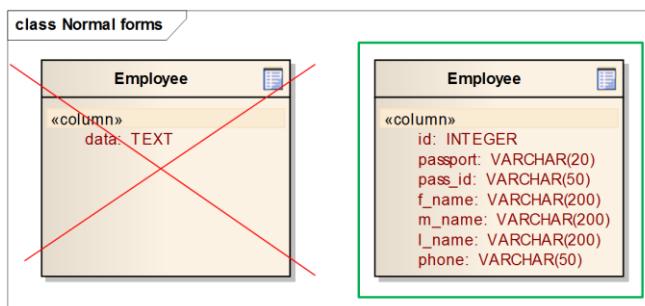
**Нормальная форма** (НФ, normal form) – ограничение схемы БД, вводимое с целью устранения определённых нежелательных свойств при выполнении реляционных операций.

#### Первая нормальная форма (1НФ)

Отношение находится в 1НФ, если все его атрибуты являются атомарными, т.е. не имеют компонентов.

Атрибут будет считаться атомарным, если в предметной области не существует операции, для выполнения которой понадобилось бы извлечь часть атрибута.

На рисунке слева представлена схема ненормализованного (до 1НФ) отношения, справа – нормализованного.



Внимание! Атомарность должна соблюдаться на уровне БД, т.е. с помощью БД не должно выполняться никаких операций над частью поля.

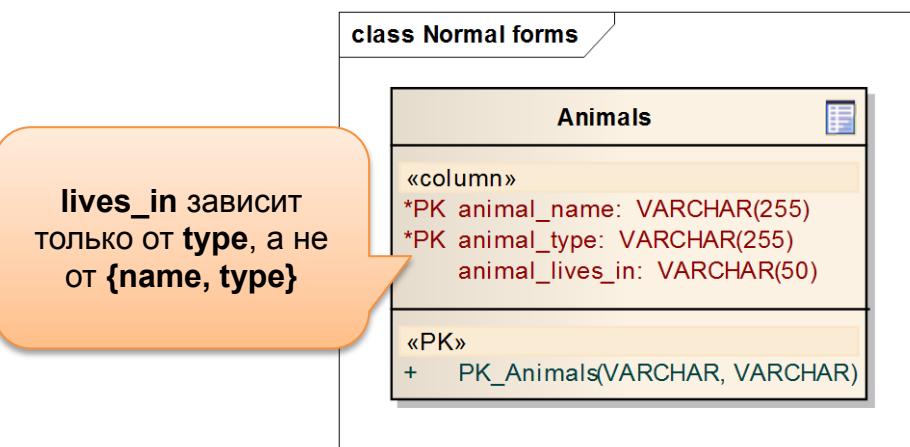
Поэтому, например, хранение в поле сериализованных данных, которые всегда добавляются, обновляются, удаляются и извлекаются ЦЕЛИКОМ, хоть и не является признаком хорошего тона, но имеет право на существование.

| umeta_id | user_id | meta_key        | meta_value                          |
|----------|---------|-----------------|-------------------------------------|
| 11       | 1       | wp_user_level   | 10                                  |
| 10       | 1       | wp_capabilities | a:1:{s:13:"administrator";s:1:"1";} |

#### Вторая нормальная форма (2НФ)

Отношение находится во 2НФ, если оно находится в 1НФ, и при этом любой атрибут, не входящий в состав ПК, функционально полно зависит от ПК.

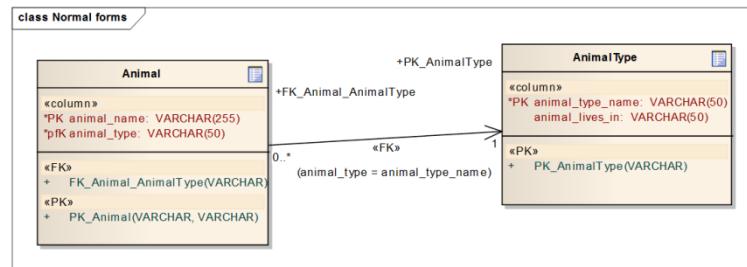
Представим, что у нас есть следующее отношение:



В силу человеческой или программной ошибки очень легко получить вот такой бред:

| animal_name | animal_type | animal_lives_in |
|-------------|-------------|-----------------|
| Пушок       | Кот         | Аквариум        |
| Йааааазь!   | Рыбина      | На диване       |

Если мы исправим модель БД (приведём ко 2НФ), всё будет хорошо:

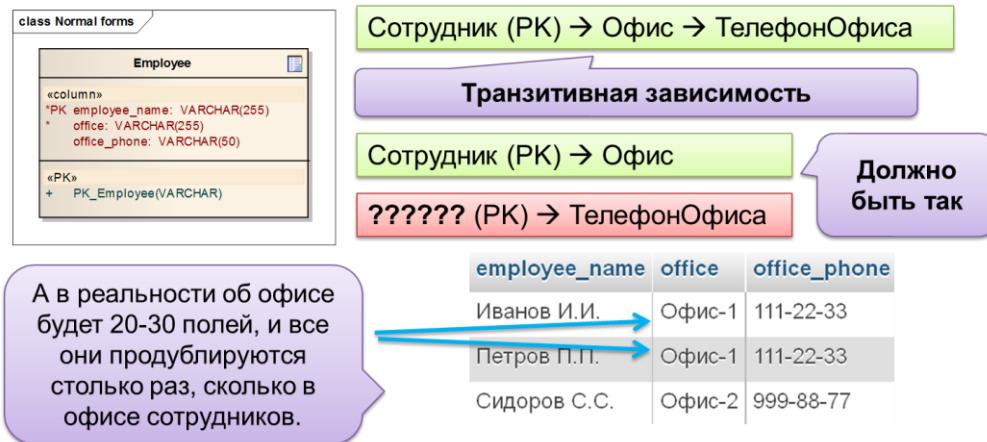


| animal_name | animal_type | animal_type_name | animal_lives_in |
|-------------|-------------|------------------|-----------------|
| Пушок       | Кот         | Кот              | На диване       |
| Йааааазь!   | Рыбина      | Рыбина           | Аквариум        |

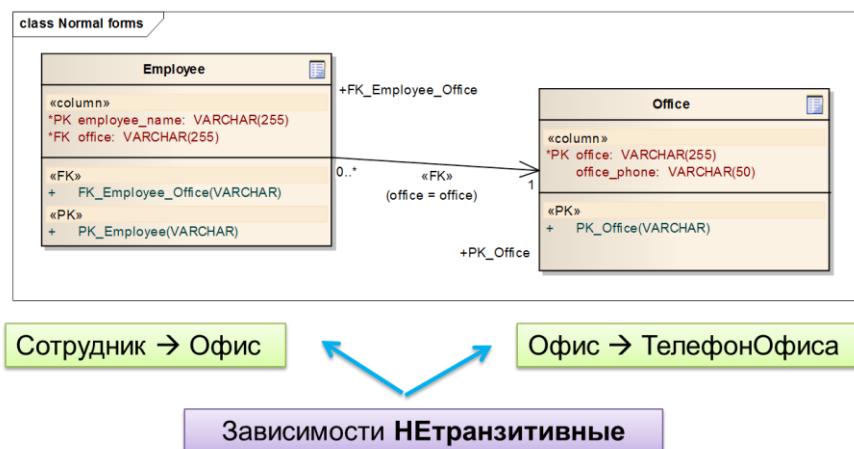
### Третья нормальная форма (3НФ)

Отношение находится в 3НФ, если оно находится во 2НФ и при этом любой его неключевой атрибут нетранзитивно (напрямую) зависит от первичного ключа.

На практике нарушение 3НФ проще всего отследить ещё и по бессмысленному дублированию данных в разных строках.



Приведём отношение к 3НФ:



С бессмысленным дублированием данных об офисе (напомним – могут быть десятки столбцов) теперь тоже всё хорошо – дублирования нет:

| employee_name | office |
|---------------|--------|
| Иванов И.И.   | Офис-1 |
| Петров П.П.   | Офис-1 |
| Сидоров С.С.  | Офис-2 |

| office | office_phone |
|--------|--------------|
| Офис-1 | 111-22-33    |
| Офис-2 | 999-88-77    |

И тут ещё 10-20 полей

### 3.3.4. Нормальные формы высоких порядков

Приведение отношения к нормальным формам высоких порядков требуется в достаточно редких, хорошо обоснованных случаях.

Не страшно, если вы не запомните эти формы. Главное – понять их, знать о них, представлять решаемые с их помощью задачи.

#### Нормальная форма Бойса-Кодда (НФБК)

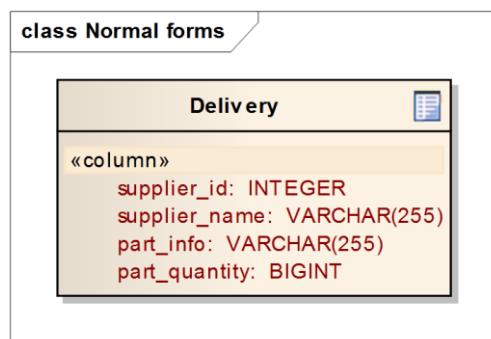
Отношение находится в НФБК, когда детерминанты всех функциональных зависимостей являются потенциальными ключами.

**Детерминант** – любой атрибут, от которого функционально-полно зависит некоторый другой атрибут. То, что стоит «слева от стрелочки» в выражениях вида  $N \rightarrow M$  (здесь  $N$  – детерминант). Он может быть сложным:  $\{A, B\} \rightarrow C$ .

Теперь ещё проще, «на пальцах»: отношение находится в ЗНФ, но НЕ находится в НФБК, если оно имеет несколько составных потенциальных ключей, у которых есть общие атрибуты.

Например, Требуется хранить данные о поставках деталей:

- Наименования поставщиков являются уникальными.
- Каждый поставщик имеет свой уникальный номер.
- Поставщик не поставляет дважды одну и ту же деталь.



Рассмотрим возможные ключи и наличие зависимостей.

Возможные ключи:

- {supplier\_id, part\_info}
- {supplier\_name, part\_info}

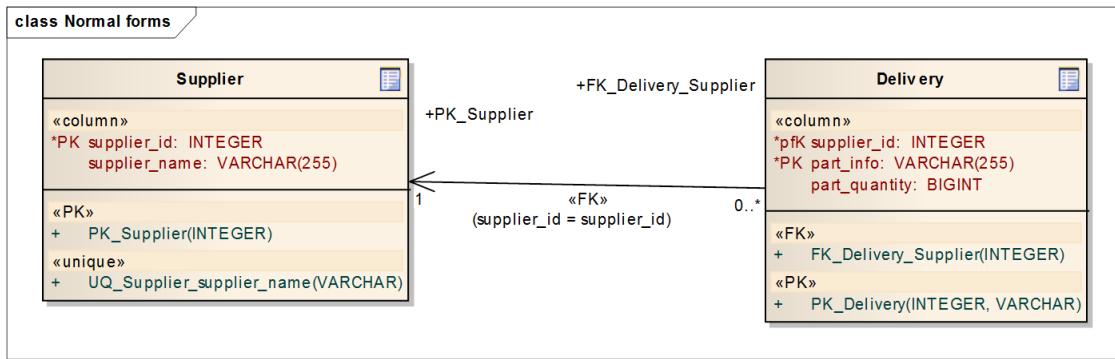
Зависимости:

- supplier\_id → supplier\_name
- supplier\_name → supplier\_id

- {supplier\_id, part\_info} → part\_quantity
- {supplier\_name, part\_info} → part\_quantity
- {supplier\_id, part\_info} → supplier\_name
- {supplier\_name, part\_info} → supplier\_id

Детерминанты **supplier\_id** и **supplier\_name** НЕ являются потенциальными ключами.

## Изменим схему БД:



Детерминанты всех функциональных зависимостей являются потенциальными ключами:

- supplier\_id → supplier\_name
- supplier\_name → supplier\_id
- {supplier\_id, part\_info} → part\_quantity

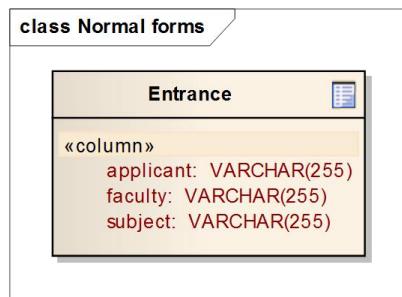
НФБК позволяет:

- УстраниТЬ избыточность хранения данных.
- Упростить операции с отдельными таблицами.
- Устраняет аномалии вставки, обновления, удаления.

## Четвёртая нормальная форма (4НФ)

Отношение находится в 4НФ, если оно находится в 3НФ или НФБК и не содержит нетривиальных многозначных зависимостей (т.е. все его зависимости являются функциональными от ключа).

Рассмотрим уже знакомый пример:



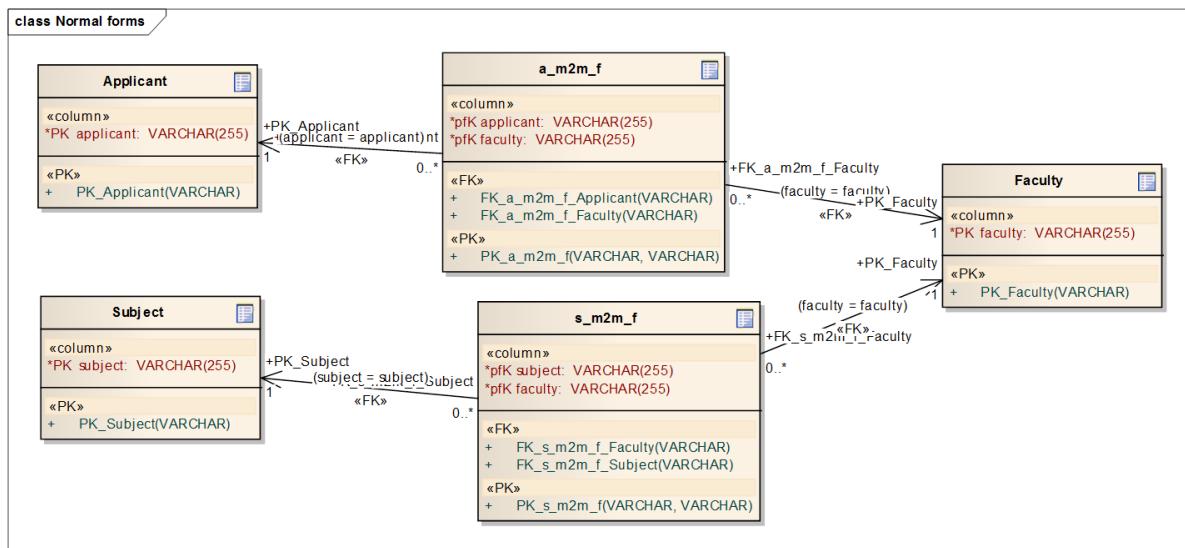
| applicant | faculty | subject      |
|-----------|---------|--------------|
| Иванов    | МатФак  | История КПСС |
| Иванов    | МатФак  | Математика   |
| Иванов    | ФизФак  | История КПСС |
| Иванов    | ФизФак  | Физика       |
| Петров    | МатФак  | История КПСС |
| Петров    | МатФак  | Математика   |

Здесь присутствует нетривиальная много-значная зависимость:

Абитуриент → → Факультет | Экзамен

~~Абитуриент → Факультет~~  
~~Абитуриент → Экзамен~~

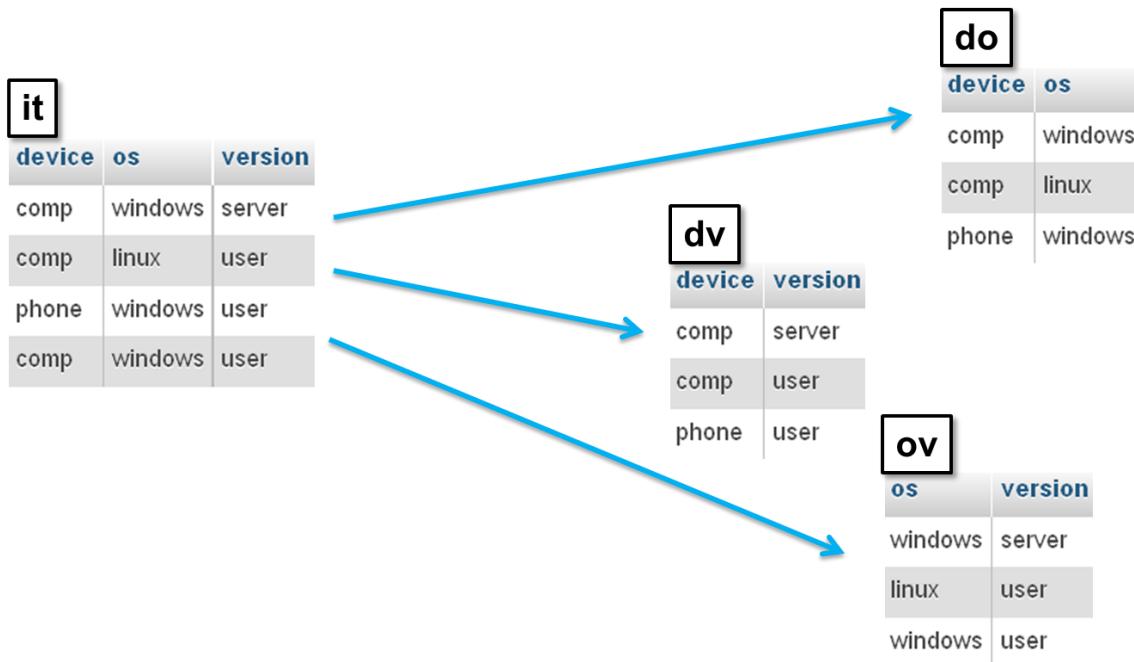
Чтобы устранить нетривиальную многозначную зависимость, нужно изменить модель БД так (здесь ТОЛЬКО функциональные зависимости):



### Пятая нормальная форма (5НФ)

Отношение находится в 5НФ, если оно находится в 4НФ и любая многозначная зависимость соединения в нём является тривиальной. Признак того, что это не выполняется: у отношения есть несколько пересекающихся составных потенциальных ключей.

Здесь не будем «копать математику», а рассмотрим пример. Возьмём отношение и получим три его проекции:



Если теперь «собрать назад» исходную таблицу с помощью попарных JOIN'ов, получится:

```
SELECT * from `dv` join `do` on
`dv`.`device`=`do`.`device`
```

| device | version | device | os      |
|--------|---------|--------|---------|
| comp   | server  | comp   | linux   |
| comp   | server  | comp   | windows |
| comp   | user    | comp   | linux   |
| comp   | user    | comp   | windows |
| phone  | user    | phone  | windows |

```
SELECT * from `dv` join `ov` on
`dv`.`version`=`ov`.`version`
```

| device | version | os      | version |
|--------|---------|---------|---------|
| comp   | user    | linux   | user    |
| phone  | user    | linux   | user    |
| comp   | server  | windows | server  |
| comp   | user    | windows | user    |
| phone  | user    | windows | user    |

```
SELECT * from `do` join `ov` on
`do`.`os`=`ov`.`os`
```

| device | os      | os      | version |
|--------|---------|---------|---------|
| comp   | linux   | linux   | user    |
| comp   | windows | windows | server  |
| comp   | windows | windows | user    |
| phone  | windows | windows | server  |
| phone  | windows | windows | user    |

Итак, мы получили (нетривиальная многозначная зависимость соединения – после JOIN'a таблиц появляются лишние строки):

| device | version | device | os      |
|--------|---------|--------|---------|
| comp   | server  | comp   | linux   |
| comp   | server  | comp   | windows |
| comp   | user    | comp   | linux   |
| comp   | user    | comp   | windows |
| phone  | user    | phone  | windows |

| device | version | os      | version |
|--------|---------|---------|---------|
| comp   | user    | linux   | user    |
| phone  | user    | linux   | user    |
| comp   | server  | windows | server  |
| comp   | user    | windows | user    |
| phone  | user    | windows | user    |

| device | os      | os      | version |
|--------|---------|---------|---------|
| comp   | linux   | linux   | user    |
| comp   | windows | windows | server  |
| comp   | windows | windows | user    |
| phone  | windows | windows | server  |
| phone  | windows | windows | user    |

А было:

| device | os      | version |
|--------|---------|---------|
| comp   | windows | server  |
| comp   | linux   | user    |
| phone  | windows | user    |
| comp   | windows | user    |

А вдруг сработает «тройной JOIN»?!

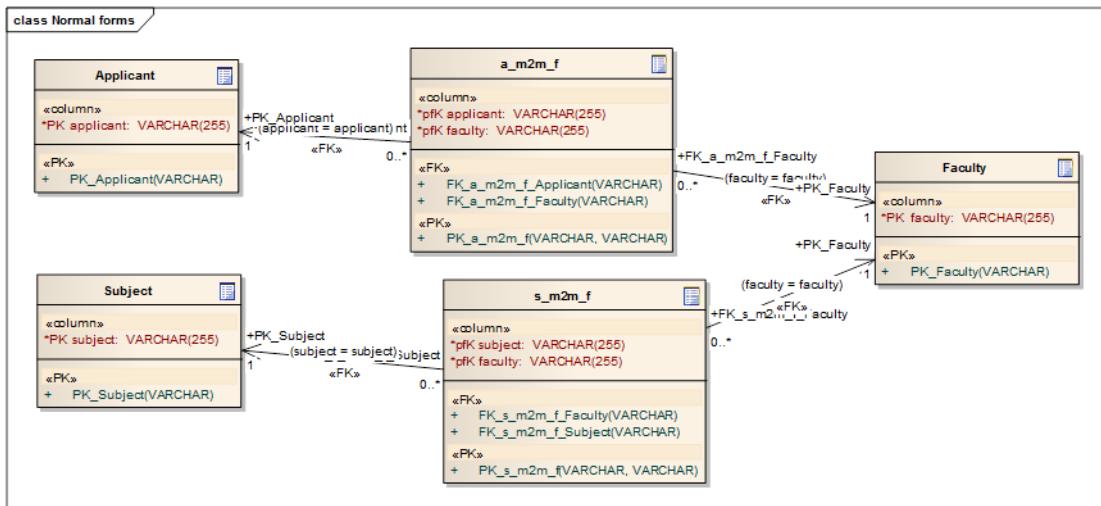
```
SELECT distinct `dv`.device, `ov`.`os`, `dv`.version from `dv` join `do` on
`dv`.device=`do`.device join `ov` on `do`.os=`ov`.os
```

Ожидали
4
Получили
5

| device | os      | version |
|--------|---------|---------|
| comp   | windows | server  |
| comp   | linux   | user    |
| phone  | windows | user    |
| comp   | windows | user    |

| device | os      | version |
|--------|---------|---------|
| comp   | linux   | server  |
| comp   | windows | server  |
| comp   | linux   | user    |
| comp   | windows | user    |
| phone  | windows | user    |

Что делать? Декомпозировать отношение подобным образом, как мы делали это в случае 4НФ: разнести атрибуты в отдельные отношения, между которыми установить связи «многие ко многим» и, если нужно, наложить ограничения с помощью триггеров.



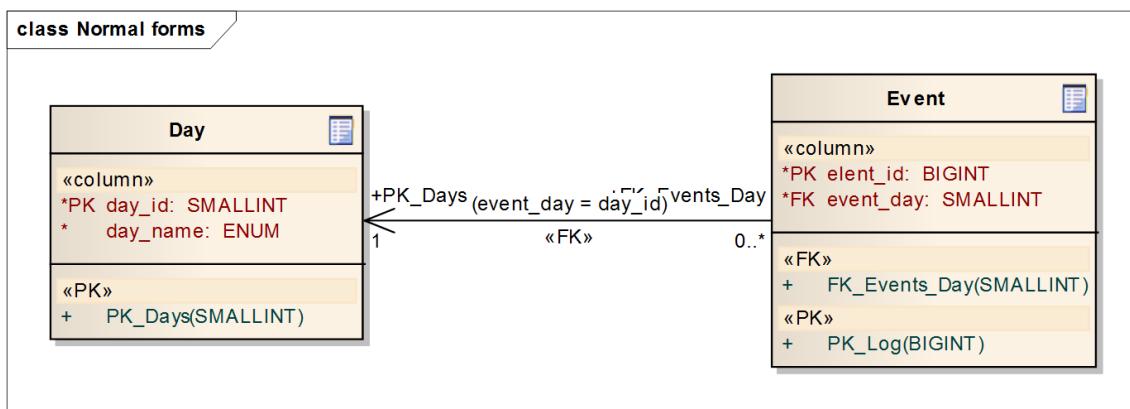
### Доменно-ключевая нормальная форма (ДКНФ)

Отношение находится в ДКНФ, если его структура не допускает аномалий вставки, обновления и удаления, а также позволяет контролировать значения атрибутов там, где это имеет смысл.

Иными словами:

- Проведите связи и укажите каскадные операции.
- Правильно используйте типы данных.

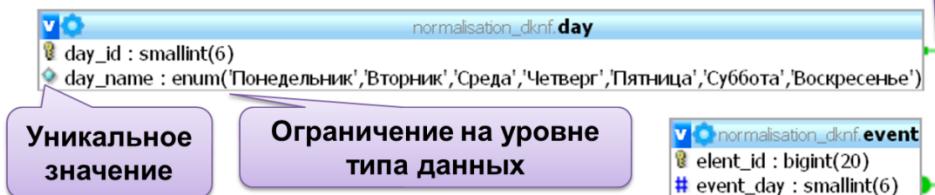
Пример: есть некоторый лог, в котором в т.ч. указывается день, в который произошло событие.



Как здесь обеспечивается ДКНФ:

| Column           | Type  | Collation       | Attributes | Null | Default |
|------------------|---|-----------------|------------|------|---------|
| <u>day_id</u>    | smallint(6)   |                 |            | No   | None    |
| <u>day_name</u>  | enum('Понедельник', 'Вторник', 'Среда', 'Четверг', 'Пятница', 'Суббота', 'Воскресенье') | utf8_general_ci |            | No   | None    |
| Column           | Type  | Collation       | Attributes | Null | Default |
| <u>event_id</u>  | bigint(20)  |                 |            | No   | None    |
| <u>event_day</u> | smallint(6)   |                 |            | No   | None    |

Связь с запретом  
каскадных операций



## Шестая нормальная форма (6НФ)

Отношение находится в 6НФ в случае, если оно находится в 5НФ, и его проекции не приводят к потере «temporalных» (временных) данных.



Проекции этого отношения с последующей «сборкой» JOIN'ом дадут (информация о том, что Иванов И.И. приходил в этот день в кабинет дважды, утеряна.):

| event_date | event_info                          |
|------------|-------------------------------------|
| 2012-03-01 |                                     |
|            | Иванову И.И. выдан ключ от кабинета |
|            | Иванов И.И. сдал ключ от кабинета   |
| 2012-03-01 | Иванову И.И. выдан ключ от кабинета |
| 2012-03-01 | Иванов И.И. сдал ключ от кабинета   |

Что делать? Либо декомпозировать отношение так, как это было в случае 4НФ, либо добавлять в отношение атрибуты (например, идентификатор события).



### 3.3.5. Краткий справочник по нормальным формам

- 1НФ – все атрибуты отношения атомарны.
- 2НФ – 1НФ + нет частичных функциональных зависимостей.
- 3НФ – 2НФ + нет транзитивных зависимостей неключевых атрибутов от ключа.
- НФБК – 3НФ + детерминанты всех функциональных зависимостей являются потенциальными ключами.
- 4НФ – 3НФ + нет нетривиальных многозначных зависимостей.
- 5НФ – 4НФ + восстановление из проекций воссоздаёт исходное отношение.
- ДКНФ – отношение не имеет аномалий модификации.
- 6НФ – 5НФ + проекции сохраняют временные данные.

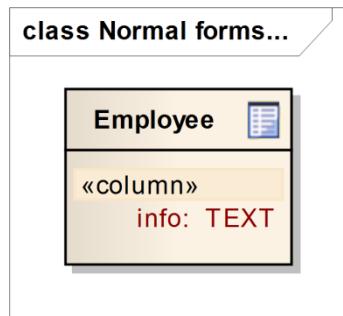
### 3.3.6. Пример применения нормализации

В большинстве случаев нормализацию завершают на 3НФ.

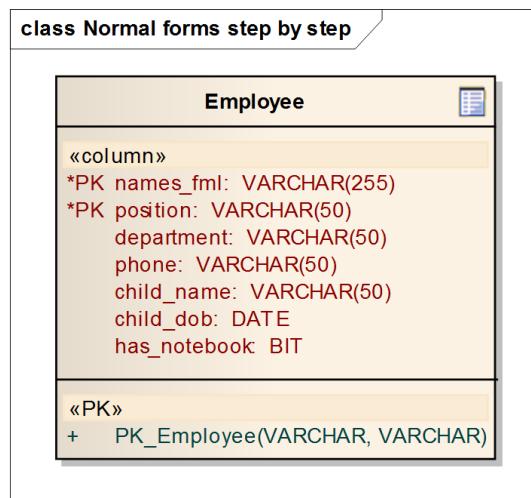
Через полгода реальной разработки БД человек физически не может придумать модель, противоречащую 3НФ, не сломав себе мозг.

#### Нормальные формы по шагам

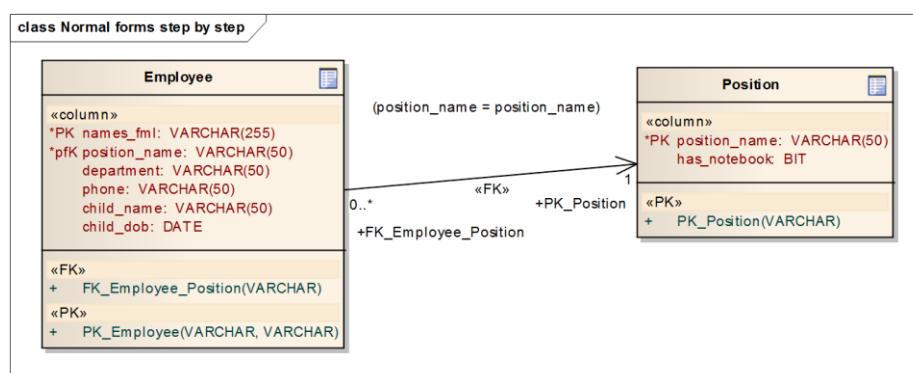
1. Ужасное отношение.



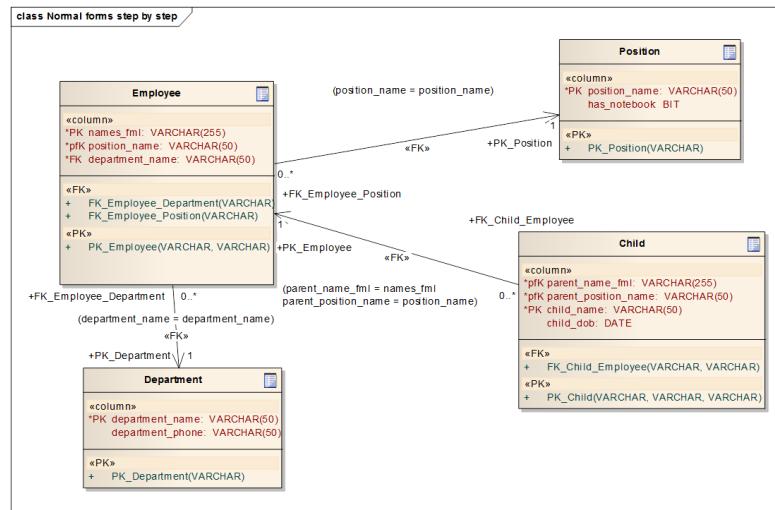
2. Приводим к 1НФ, атомизируем атрибуты:



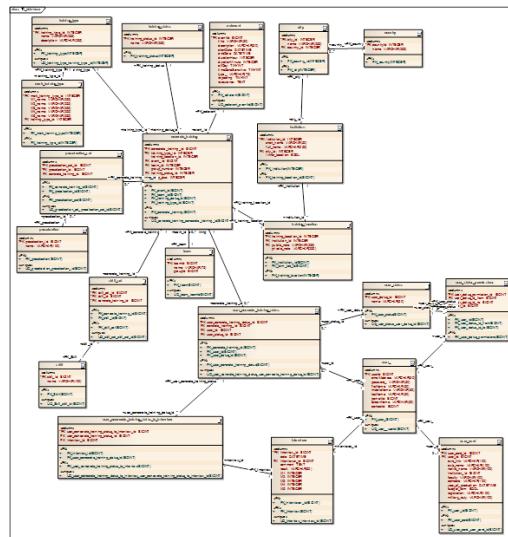
3. У начальника есть ноутбук, у подчинённого – нет. Убираем частичную ФЗ. Получаем 2НФ.



#### 4. Убираем транзитивные ФЗ, получаем ЗНФ.



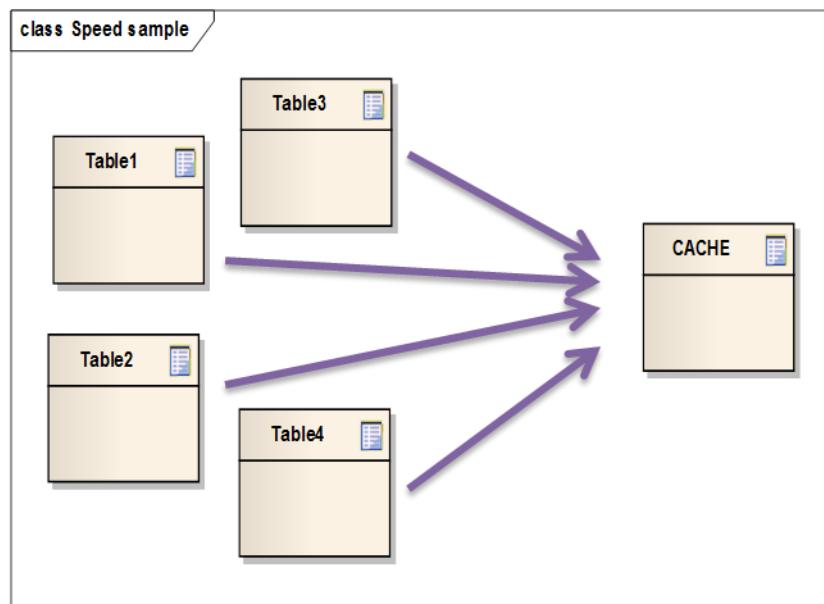
#### 5. Добавляем забытые атрибуты, повторяем шаги 1-4 ☺.



### 3.4. Денормализация

**Денормализация** (denormalization) – процесс приведения отношения к состоянию, нарушающему те или иные нормальные формы.

Денормализация выполняется, в основном, для создания «кэширующих таблиц», некоторые операции с которыми могут выполняться намного быстрее, чем с набором исходных таблиц.



## Тема 4: Проектирование баз данных

### 4.1. Оценка сложности БД

#### 4.1.1. Быстрая оценка

Перед началом проектирования всегда необходимо определить, сколько времени и иных ресурсов нам понадобится.

Существует множество способов оценки трудозатрат, но в контексте проектирования БД мы можем использовать простую таблицу.

**ВАЖНО:** это ПРИБЛИЗИТЕЛЬНАЯ оценка. Для точной оценки нужно использовать более сложные методы.

Все дальнейшие примеры мы будем рассматривать в контексте задания: «Спроектировать БД файлообменного сервиса».

Оценим сложность создания БД, используя следующие критерии:

| №            | Параметр                                    | Оценка  | Коэффициент | Итог        |
|--------------|---|---------|-------------|-------------|
| 1            | Количество таблиц                           | 30      | 2           | 60          |
| 2            | Понимание предметной области                | 8 из 10 | -3          | -24         |
| 3            | Наличие средств проектирования              | 8 из 10 | -2.5        | -20         |
| 4            | Доработки в процессе проектирования         | 7 из 10 | 2.5         | 17.5        |
| 5            | Специфические технологии и требуемые знания | 9 из 10 | 3           | 27          |
| 6            | Квалификация команды                        | 9 из 10 | -4          | -36         |
| 7            | Загруженность команды другими задачами      | 7 из 10 | 4           | 28          |
| <b>ИТОГО</b> |   |         |             | <b>52.5</b> |

Мы только что получили число 52.5. О чём это вам говорит? **НИ. О. ЧЁМ.** Нужны исследования, чтобы спроектировать это число на что-то понятное.

Пример результатов подобных исследований представлен в следующей таблице (внимание: это именно ПРИМЕР, т.е. не пытайтесь «напрямую» применить его для оценки ВАШИХ трудозатрат, т.к. вам нужно знать ВАШУ производительность, ВАШУ квалификацию и т.п.):

| Коэффициент сложности | Описание                            | Примерное время работы команды                                  |
|-----------------------|-------------------------------------|---|
| < 0                   | Лабораторная работа третьего курса. | Мы этим не занимаемся.  |
| 0-10                  | Сайт фирмы «Рога и копыта».         | Один человек, пара часов.                                       |
| 10-40                 | Средняя БД.                         | Вся команда за день или один человек 1-2 дня.                   |
| <b>40-100</b>         | <b>Интересная БД.</b>               | <b>Вся команда 1-2 дня или один человек 3-4 дня.</b>            |
| 100-500               | Сложная БД.                         | Вся команда 5-10 дней.  |
| 500-5000              | Очень сложная БД.                   | Вся команда 3-4 недели.   |
| > 5000                | Неподъёмный проект.                 | Наших сил не хватит, нужно обсуждать и искать решения отдельно. |

*«Проект-менеджерское отступление»: как показала практика, у одного человека, занимавшегося проектированием данной БД на создание рассмотренной здесь модели БД ушло чуть больше дня. Учитывая, что модель ещё не закончена, оценка оказалась вполне верной.*



## 4.2. Проектирование БД на инфологическом уровне

Вспомним определение инфологического уровня моделирования.

### Инфологический уровень

Описание предметной области,  
нет привязки к СУБД,  
предназначено «для людей».

#### Описывать модель здесь можно

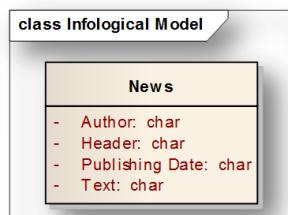
«Словами»  
(просто в виде  
списков)

«Табличками» (в  
Word или подобных  
редакторах)

Специальными  
графическими  
нотациями (UML и т.п.)

- Новости:
- Заголовок
  - Автор
  - Дата публикации
  - Текст

| Новости | Заголовок | Автор | Дата публикации | Текст |
|---------|-----------|-------|-----------------|-------|
|         | Текст     | Текст | Дата            | Текст |



**Главная задача** – выявление всех сущностей и атрибутов. Сразу это не получится, процесс будет итерационным. Здесь пока можно не прописывать сущности для организации связей М-М, но можно указать, что те или иные поля будут использоваться для организации связей.

#### Сущности и атрибуты

1. Категории файлов:
  - a. Идентификатор категории.
  - b. Название категории.
  - c. Возрастные ограничения (ссылка).
2. Файлы на сервере:
  - a. Идентификатор файла.
  - b. Идентификатор категории, к которой относится файл (ссылка).
  - c. Размер (в байтах).
  - d. Дата добавления.
  - e. Срок хранения (до какой даты и какого времени).
  - f. Исходное имя (без расширения).
  - g. Исходное расширение.
  - h. Имя на сервере (хэш).
  - i. Контрольная сумма (sha1-хэш).
  - j. Параметры доступа (ссылка).
  - k. Количество скачиваний файла.
  - l. Возрастные ограничения (ссылка).
  - m. Ссылка на удаление (для незарегистрированных пользователей).

3. Параметры доступа к файлу:
  - a. Идентификатор параметра.
  - b. Наименование параметра.
  - c. Описание параметра.
  - d. Создан пользователем (признак того, что данный параметр не является «встроенным», а создан пользователем).
4. Возрастные ограничения:
  - a. Идентификатор ограничения.
  - b. Минимальный возраст (в годах).
  - c. Название ограничения.
  - d. Описание ограничения.
5. Комментарии (плоская структура, т.е. нельзя комментировать комментарии):
  - a. Идентификатор комментария.
  - b. К какому файлу (ссылка).
  - c. Текст комментария.
  - d. Кто оставил комментарий (ссылка).
  - e. Дата и время добавления комментария.
  - f. Выставленная оценка (комментируя файл, можно не только писать текст, но и выставлять файлу оценку).
6. Пользователи:
  - a. Идентификатор пользователя.
  - b. Логин.
  - c. Пароль.
  - d. E-mail.
  - e. Дата и время регистрации.
  - f. Дата рождения.
  - g. Сколько файлов пользователь закачал.
  - h. Сколько файлов пользователь скачал.
  - i. Сколько комментариев оставил пользователь.
  - j. Бонус по скорости.
  - k. Бонус по количеству закачиваемого.
  - l. Бонус по количеству скачиваемого.
  - m. Блокировка (бан) (ссылка).
  - n. До какой даты и какого времени заблокирован.
7. Роли (пользователь может находиться в нескольких ролях):
  - a. Идентификатор роли.
  - b. Название роли.
  - c. Ограничения по объёму закачиваемых файлов.
  - d. Ограничения по объёму скачиваемых файлов.
  - e. Ограничения по количеству закачиваемых файлов.
  - f. Ограничения по количеству скачиваемых файлов.
  - g. Ограничения по скорости скачивания.
8. Права ролей (правами обладают только роли, напрямую пользователям права давать нельзя):
  - a. Идентификатор права.
  - b. Название права.
  - c. Описание права.
9. Файлы, закачанные данным пользователем:
  - a. Идентификатор пользователя.
  - b. Идентификатор файла.
10. Файлы, скачанные данным пользователем:

- a. Идентификатор пользователя.
- b. Идентификатор файла.

11. Лог (хранит действия за последний месяц, потом они переносятся в архив лога):

- a. Идентификатор записи лога.
- b. Идентификатор пользователя (ссылка).
- c. Идентификатор действия (ссылка).
- d. Идентификатор файла (ссылка).
- e. Дата и время выполнения действия.
- f. Параметры действия.

12. Архив лога:

- a. Идентификатор записи.
- b. Идентификатор пользователя (ссылка).
- c. Идентификатор действия (ссылка).
- d. Идентификатор файла (ссылка).
- e. Дата и время выполнения действия.
- f. Параметры действия.

13. Чёрный список IP-адресов:

- a. IP-адрес в формате IPv4.
- b. IP-адрес в формате IPv6.
- c. Дата и время, до которых действительна блокировка.
- d. Причина блокировки (ссылка).

14. Причины блокировок:

- a. Идентификатор причины.
- b. Название причины.
- c. Описание причины.

15. Страницы сайта:

- a. Идентификатор страницы.
- b. Ссылка на родительскую страницу.
- c. Название страницы (для отображения на самой странице).
- d. Имя страницы в меню и на карте сайта.
- e. Заглавие страницы (HTML-тег TITLE).
- f. Ключевые слова (HTML-тег meta ... keywords).
- g. Описание страницы (HTML-тег meta ... description).
- h. Текстовое наполнение страницы.
- i. Основной шаблон страницы.
- j. Основной обработчик страницы.

16. Ссылки на скачивание файлов:

- a. Идентификатор ссылки.
- b. Идентификатор файла (ссылка).
- c. Ссылка (sha1-хэш).
- d. Срок действия ссылки (дата и время, после которых ссылка считается недействительной).
- e. IP-адрес, для которого действительна ссылка (в формате IPv4).
- f. IP-адрес, для которого действительна ссылка (в формате IPv6).
- g. Хэш для размещения в cookie (для дополнительной защиты; не должен совпадать с хэшем самой ссылки).
- h. Пароль на скачивание.

17. Интенсивность скачиваний:

- a. Идентификатор скачивания.
- b. Идентификатор ссылки на скачивание (ссылка).
- c. Дата и время, когда было произведено скачивание.

18. Информация о ресурсе (все поля кэшируются на основе агрегирования данных из других таблиц):

- a. Всего зарегистрированных пользователей.
- b. Пользователей зарегистрировалось сегодня.
- c. Всего закачано файлов.
- d. Файлов закачано сегодня.
- e. Общий объём закачанных файлов.
- f. Объём файлов, закачанных сегодня.
- g. Всего скачано файлов.
- h. Файлов скачано сегодня.
- i. Общий объём скачанных файлов.
- j. Объём файлов, скачанных сегодня.

19. Группы пользователей:

- a. Идентификатор группы.
- b. Владелец (создатель) группы (ссылка).
- c. Название группы.
- d. Описание группы.

Представленный выше список является предельно общим описанием сути проектируемой БД. Он специально оставлен в виде «word'овского списка», чтобы всегда иметь возможность обсудить ситуацию с заказчиком, не прибегая к графическим нотациям (которые представитель заказчика может просто напросто не понимать или понимать не до конца или неверно). Однако, если со стороны заказчика с нами общается человек, достаточно разбирающийся в графических нотациях, более выгодно можно представить инфологический уровень данной БД следующим образом с использованием UML.

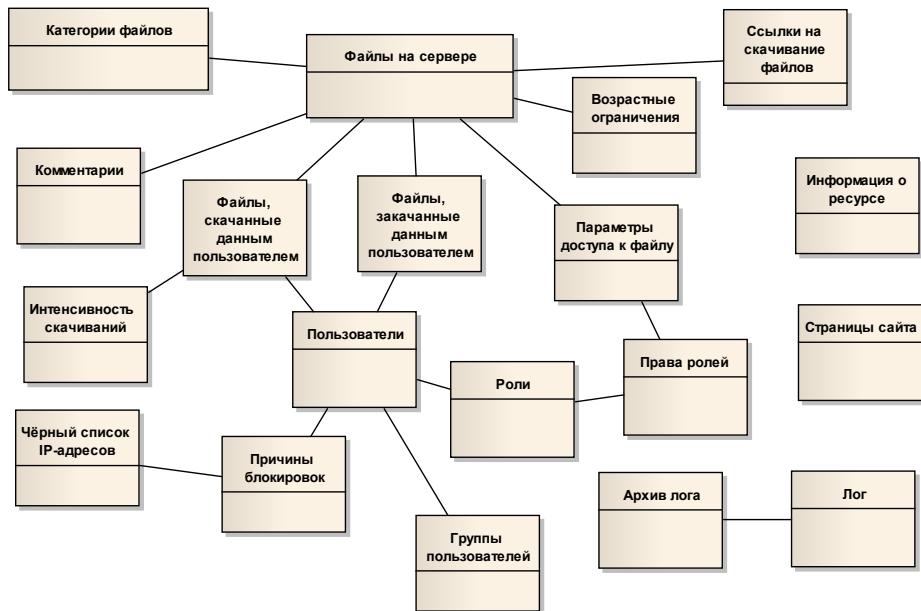
## ПОЧЕМУ WORD?!

Быстро, просто.

Удобно обсуждать с заказчиком (любым).

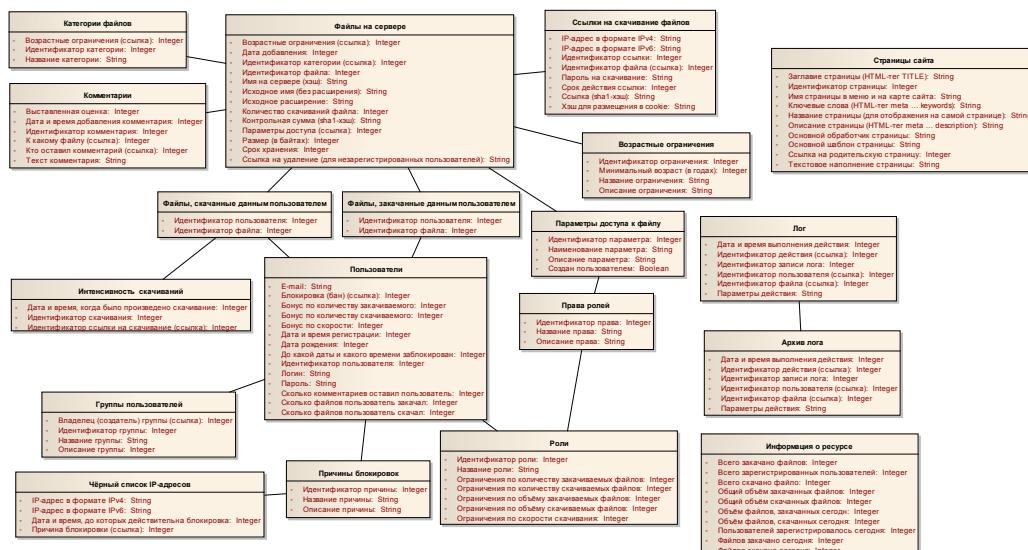
Удобно дорабатывать.

Можно прописывать комментарии, которые сразу же видны (не нужно открывать свойства элементов).



На начальных этапах не имеет смысла сразу же прописывать на такой UML-диаграмме атрибуты сущностей, т.к. в процессе построения и обсуждения такой диаграммы может возникнуть множество новых идей, могут быть обнаружены какие-то проблемы и т.д. Всё это приведёт к тому, что модель (и вышеприведённый список сущностей и атрибутов) многократно поменяется.

Если всё же прописать атрибуты (детализировать модель), она примет следующий вид.



Ещё одним хорошим решением является печать модели на листах формата А3/A2/A1 и т.д. Да, это достаточно дорого, но такую распечатку можно проводить после внесения масштабных изменений.

### 4.3. Проектирование БД на даталогическом уровне

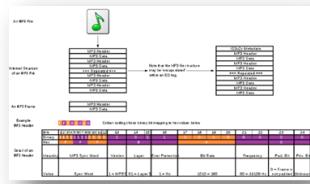
Вспомним определение даталогического уровня моделирования.

#### Даталогический уровень

Модель предметной области в привязке к СУБД определённого вида или к конкретной СУБД. Описывает способ организации данных безотносительно их физического размещения

#### Описывать модель здесь можно

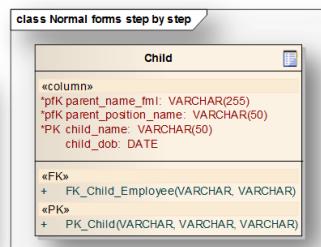
Спецификациями форматов данных



Собственными средствами формата (XSD в XML)



Специальными графическими нотациями (UML и т.п.)



После того, как все высокоуровневые требования, отражённые на модели инфологического уровня, согласованы с заказчиком, имеет смысл приступить к созданию проекции инфологической модели на даталогический уровень, т.е. описать модель в терминах определённого вида БД, или даже в синтаксисе конкретной СУБД.

Но перед тем, как создавать модель, необходимо ещё раз проанализировать имеющуюся инфологическую модель: на этот раз мы будем анализировать её с точки зрения «как реализовать это в виде БД», оставляя важные для проектирования БД и создания кода приложения комментарии.

Для удобства оформим нашу работу в виде таблицы. И, прежде чем приступить к работе, примем несколько соглашений, распространяющихся на всю работу по созданию даталогической модели:

- В таблице с описанием таблиц и полей БД тип данных указывается явно, если из названия полей и здравого смысла не очевидно, что нужно использовать «обычные» текстовые или числовые типы данных.
- Все типы данных при создании полей таблиц БД формируются из расчёта совместимости с MySQL версии 5.5 и выше.
- При именовании таблиц и полей используется ТОЛЬКО нижний регистр.
- Слова в именах таблиц и полей отделены друг от друга символом «\_».
- В именах таблиц существительные используются в единственном числе (file, а не files). В именах полей допускается множественное число.
- Имена полей начинаются с префиксов из первых букв названия таблицы.
- Все поля, содержащие дату и/или время, имеют тип INTEGER и хранят UNIXTIME-значения.

| Таблица   | Поле   | Тип             | Комментарий  |
|---|--|-----------------|--|
| Категории файлов  | Идентификатор категории                                      |                 |  |
|   | Название категории   |                 |  |
|   | Возрастные ограничения (ссылка)                              |                 |  |
| Файлы на сервере  | Идентификатор файла  | UNSIGNED BIGINT |  |
|   | Идентификатор категории, к которой относится файл (ссылка)   |                 |  |
|   | Размер (в байтах)  |                 |  |
|   | Дата добавления  |                 |  |
|   | Срок хранения (до какой даты и какого времени)               |                 |  |
|   | Исходное имя (без расширения)                                |                 |  |
|   | Исходное расширение  |                 |  |
|   | Имя на сервере (хэш)   |                 |  |
|   | Контрольная сумма (sha1-хэш)                                 |                 |  |
|   | Параметры доступа (ссылка)                                   |                 |  |
|   | Количество скачиваний файла                                  |                 |  |
|   | Возрастные ограничения (ссылка)                              |                 |  |
| Параметры доступа к файлу   | Ссылка на удаление (для не зарегистрированных пользователей) |                 |  |
|   | Идентификатор параметра                                      |                 |  |
|   | Наименование параметра                                       |                 |  |
|   | Описание параметра   |                 |  |
| Возрастные ограничения  | Создан пользователем   |                 | Признак того, что данный параметр не является «встроенным», а создан пользователем |
|   | Идентификатор ограничения                                    |                 |  |
|   | Минимальный возраст (в годах)                                |                 |  |
|   | Название ограничения   |                 |  |
|   | Описание ограничения   |                 |  |
| Комментарии (плоская структура, т.е. нельзя комментировать комментарии) | Идентификатор комментария                                    | UNSIGNED BIGINT |  |
|   | К какому файлу (ссылка)                                      |                 |  |
|   | Текст комментария  |                 |  |
|   | Кто оставил комментарий (ссылка)                             |                 |  |
|   | Дата и время добавления комментария                          |                 |  |
|   | Выставленная оценка  |                 | комментируя файл, можно не только писать текст, но и выставлять файлу оценку       |
| Пользователи  | Идентификатор пользователя                                   | UNSIGNED BIGINT |  |
|   | Логин  |                 |  |
|   | Пароль   |                 |  |
|   | E-mail   |                 |  |
|   | Дата и время регистрации                                     |                 |  |
|   | Дата рождения  |                 |  |
|   | Сколько файлов пользователь закачал                          |                 | Кэшируется из закачек.   |

|   |   |  |  |
|---|---|--|--|
|   | Сколько файлов пользователь скачал            |  | Кэшируется из скачанного.  |
|   | Сколько комментариев оставил пользователь     |  | Кэшируется из комментариев.  |
|   | Бонус по скорости                             |  |  |
|   | Бонус по количеству закачиваемого             |  |  |
|   | Бонус по количеству скачиваемого              |  |  |
|   | Бан (ссылка)                                  |  | (ссылка на причину бана; NULL – если не забанен)   |
|   | До какой даты и какого времени забанен        |  | (NULL – если не забанен)   |
| Роли (пользователь может находиться в нескольких ролях)<br>Права ролей (правами обладают только роли, напрямую пользователям права давать нельзя) | Идентификатор роли                            |  |  |
|   | Название роли                                 |  |  |
|   | Ограничения по объёму закачиваемых файлов     |  |  |
|   | Ограничения по объёму скачиваемых файлов      |  |  |
|   | Ограничения по количеству закачиваемых файлов |  |  |
|   | Ограничения по количеству скачиваемых файлов  |  |  |
|   | Ограничения по скорости скачивания            |  |  |
|   | Идентификатор права                           |  |  |
|   | Название права                                |  |  |
|   | Описание права                                |  |  |
| Файлы, закачанные или скачанные данным пользователем  | Идентификатор пользователя                    |  | Объединение двух исходных отношений («Файлы, закачанные данным пользователем» и «Файлы, скачанные данным пользователем») |
|   | Идентификатор файла                           |  |  |
| Лог (хранит действия за последний месяц, потом они переносятся в архив лога)  | Идентификатор записи лога                     | UNSIGNED BIGINT                                    |  |
|   | Идентификатор пользователя (ссылка)           |  | NULL, если это системное действие  |
|   | Идентификатор действия (ссылка)               |  | Ссылка на таблицу прав.  |
|   | Идентификатор файла (ссылка)                  |  | NULL, если операция не связана с файлами   |
|   | Дата и время выполнения действия              |  |  |
|   | Параметры действия                            |  | Сериализованный массив.  |
| Архив лога  | Идентификатор записи лога                     | 255-разрядное 36-ричное автоинкрементируемое число | Очень сильно подумать о том, как это реализовать: триггером, хранимой процедурой, на уровне приложения и т.п.            |
|   | Идентификатор пользователя (ссылка)           |  | NULL, если это системное действие  |
|   | Идентификатор действия (ссылка)               |  | Ссылка на таблицу прав.  |
|   | Идентификатор файла (ссылка)                  |  | NULL, если опера-  |

|                               |   |  |  |
|-------------------------------|---|--|--|
|                               | ка)   |  | ция не связана с файлами   |
|                               | Дата и время выполнения действия  |  |  |
|                               | Параметры действия  |  | Сериализованный массив.  |
| Чёрный список IP-адресов      | IP-адрес в формате IPv4   |  |  |
|                               | IP-адрес в формате IPv6   |  |  |
|                               | Дата и время, до которых действительна блокировка   |  |  |
|                               | Причина блокировки (ссылка)   |  |  |
| Причины блокировок            | Идентификатор причины   |  |  |
|                               | Название причины  |  |  |
|                               | Описание причины  |  |  |
| Страницы сайта                | Идентификатор страницы  |  |  |
|                               | Ссылка на родительскую страницу   |  |  |
|                               | Название страницы (для отображения на самой странице)   |  |  |
|                               | Имя страницы в меню и на карте сайта  |  |  |
|                               | Заглавие страницы (HTML-тег TITLE)  |  |  |
|                               | Ключевые слова (HTML-тег meta ... keywords)   |  |  |
|                               | Описание страницы (HTML-тег meta ... description)   |  |  |
|                               | Текстовое наполнение страницы   |  |  |
|                               | Основной шаблон страницы  |  |  |
|                               | Основной обработчик страницы  |  |  |
| Ссылки на скачивание файлов   | Идентификатор ссылки  |  |  |
|                               | Идентификатор файла (ссылка)  |  |  |
|                               | Ссылка (sha1-хэш)   |  | хэш, 240 символов, 6 sha1  |
|                               | Срок действия ссылки (дата и время, после которых ссылка считается недействительной)              |  |  |
|                               | IP-адрес, для которого действительна ссылка (в формате IPv4)                                      |  | Эти два поля не могут одновременно быть NULL (сделать триггер для проверки).           |
|                               | IP-адрес, для которого действительна ссылка (в формате IPv6)                                      |  |  |
|                               | Хэш для размещения в cookie (для дополнительной защиты; не должен совпадать с хэшем самой ссылки) |  | Хэш ссылки и cookie-хэш НЕ ДОЛЖНЫ СОВПАДАТЬ! Продумать реализацию (видимо, триггером). |
|                               | Пароль на скачивание  |  |  |
| Интенсивность скачиваний      | Идентификатор скачивания  |  |  |
|                               | Идентификатор ссылки на скачивание (ссылка)   |  |  |
|                               | Дата и время, когда было произведено скачивание   |  |  |
| Информация о ресурсе (все по- | Всего зарегистрированных пользователей  |  | Хорошо продумать обеспечение конси-  |

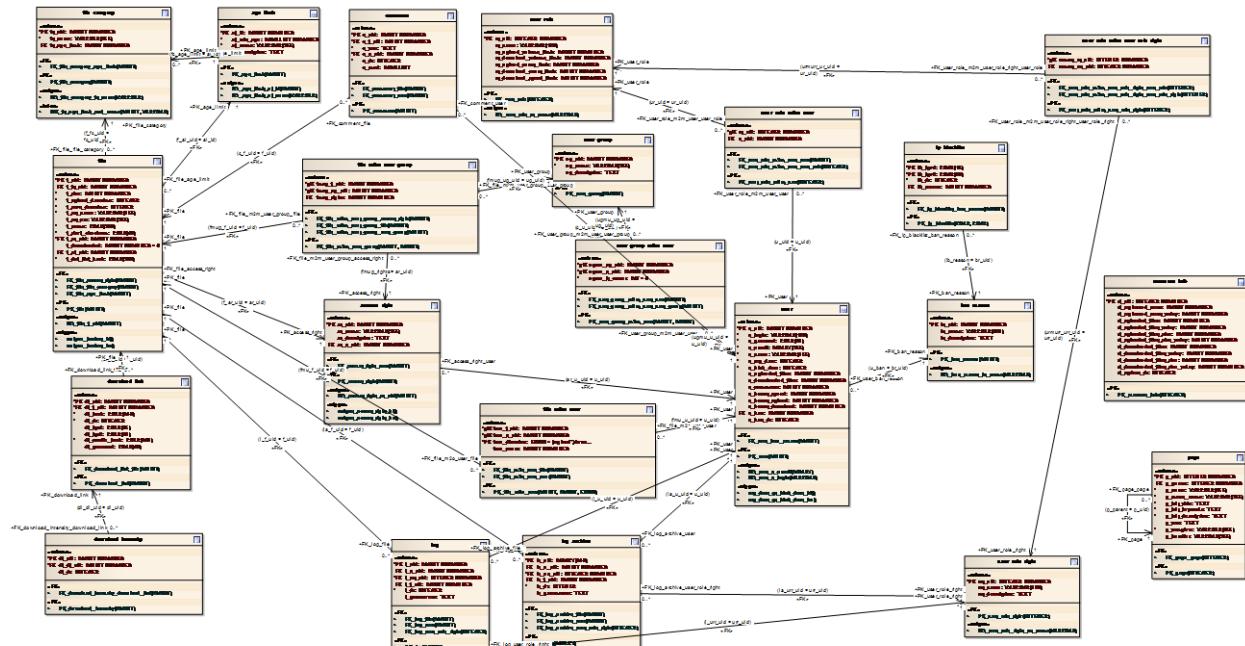
|  |  |  |                    |
|--|--|--|--------------------|
| ля кэшируются на основе агрегирования данных из других таблиц: | Пользователей зарегистрировалось сегодня |  | стентности данных! |
|  | Всего закачано файлов                    |  |                    |
|  | Файлов закачано сегодня                  |  |                    |
|  | Общий объём закачанных файлов            |  |                    |
|  | Объём файлов, закачанных сегодня         |  |                    |
|  | Всего скачано файлов                     |  |                    |
|  | Файлов скачано сегодня                   |  |                    |
|  | Общий объём скачанных файлов             |  |                    |
|  | Объём файлов, скачанных сегодня          |  |                    |
|  |  |  |                    |
| Группы пользователей   | Идентификатор группы                     |  |                    |
|  | Владелец (создатель) группы (ссылка)     |  |                    |
|  | Название группы                          |  |                    |
|  | Описание группы                          |  |                    |

Несмотря на то, что вышеприведённая таблица не слишком сильно отличается от ранее показанного списка сущностей и атрибутов, она может понадобиться при подробном документировании проекта (когда типы данных и комментарии будут прописаны подробно по всем полям всех таблиц).

Поскольку в реальности гораздо удобнее хранить описания и графическое представление в едином файле, НАСТОЯТЕЛЬНО рекомендуется писать комментарии ко ВСЕМ таблицам и ВСЕМ полям, даже ЕСЛИ сейчас вам что-то кажется понятным и самоочевидным: через некоторое время вы что-то забудете, и понятным и очевидным оно не будет уже и для вас. А для тех, кто впервые видит модель БД, уже сейчас там много непонятного и неочевидного.

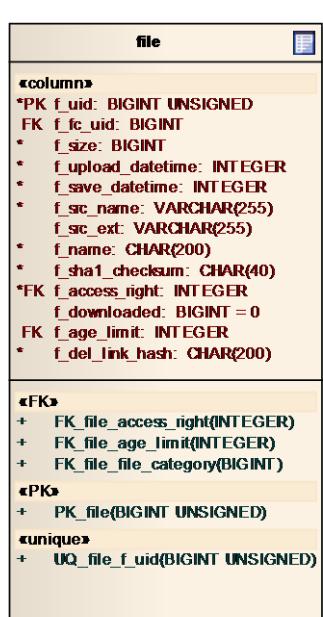
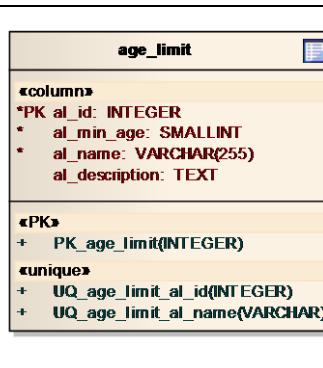
Итак, представим нашу модель графически, а затем дадим необходимые пояснения.

Как и в случае инфологического моделирования, модель получилась достаточно объёмной, а потому «на одной картинке» она сложно читается. Но один факт можно заметить сразу: таблиц в БД больше (22), чем изначально было выделено сущностей (19). Это связано с тем, что даталогический уровень уже содержит таблицы для связей «многие ко многим», а также отражает результат нормализации и иных форм адаптации модели предметной области к реальности (в т.ч. адаптации к требованию удобства использования при написании кода приложения).



Снова обратимся к табличному представлению и прокомментируем особенности той или иной таблицы БД.

| Исходная таблица | Таблица БД   | Комментарии   |
|------------------|--|---|
| Категории файлов | <p><b>file_category</b></p> <pre> «column» *PK fc_uid: BIGINT + fc_name: VARCHAR(255) FK fc_age_limit: INTEGER  «FK» + FK_file_category_age_limit(INTEGER) «PK» + PK_file_category(BIGINT) «unique» + UQ_file_category_fc_name(VARCHAR) + UQ_file_category_fc_uid(BIGINT) </pre> | Категории файлов. Например, "Музыка", "Видео", "Программы" и т.п. |

|                           |   |  |
|---------------------------|---|--|
| Файлы на сервере          |  <pre> file +--&lt;column&gt; *PK f_uid: BIGINT UNSIGNED FK f_fc_uid: BIGINT * f_size: BIGINT * f_upload_datetime: INTEGER * f_save_datetime: INTEGER * f_sc_name: VARCHAR(255) f_sc_ext: VARCHAR(255) * f_name: CHAR(200) * f_sha1_checksum: CHAR(40) *FK f_access_right: INTEGER f_downloaded: BIGINT = 0 FK f_age_limit: INTEGER * f_del_link_hash: CHAR(200)  &lt;FKs&gt; + FK_file_access_right(INTEGER) + FK_file_age_limit(INTEGER) + FK_file_file_category(BIGINT)  &lt;PKs&gt; + PK_file(BIGINT UNSIGNED)  &lt;unique&gt; + UQ_file_f_uid(BIGINT UNSIGNED) </pre> | <p>Одна из основных, наиболее часто используемых таблиц.</p> <p><b>Максимально продумать оптимизацию производительности!</b></p> <p>Файл сохраняется на сервере под именем, состоящим из пяти sha1-хешей, получаемых на основе имени файла и случайных величин.</p> <p>f_del_link_hash хранит пять sha1-хешей, получаемых на основе случайных величин. Значение этого поля НЕ ДОЛЖНО совпадать с f_name.</p> |
| Параметры доступа к файлу |  <pre> access_right +--&lt;column&gt; *PK ar_uid: INTEGER ar_name: VARCHAR(255) ar_description: TEXT ar_user_created: BIT = 1  &lt;PKs&gt; + PK_access_right(INTEGER)  &lt;unique&gt; + UQ_access_right_ar_name(VARCHAR) + UQ_access_right_ar_uid(INTEGER) </pre>   | <p>Параметры доступа к файлу. Например, "публичный", "личный", "для друзей", "для указанного круга пользователей".</p> <p>Пользователи могут создавать свои наборы прав доступа (ar_user_created == 1).</p> <p>Изначально в таблице расположено несколько «предустановленных» прав, например: «файл виден всем», «файл виден только владельцу» и т.п.</p>  |
| Возрастные ограничения    |  <pre> age_limit +--&lt;column&gt; *PK al_id: INTEGER * al_min_age: SMALLINT * al_name: VARCHAR(255) al_description: TEXT  &lt;PKs&gt; + PK_age_limit(INTEGER)  &lt;unique&gt; + UQ_age_limit_al_id(INTEGER) + UQ_age_limit_al_name(VARCHAR) </pre>  | <p>Возрастные ограничения. Наименование ограничения -- TBD, см. международную практику по маркировке фильмов, игр и т.п.</p>   |

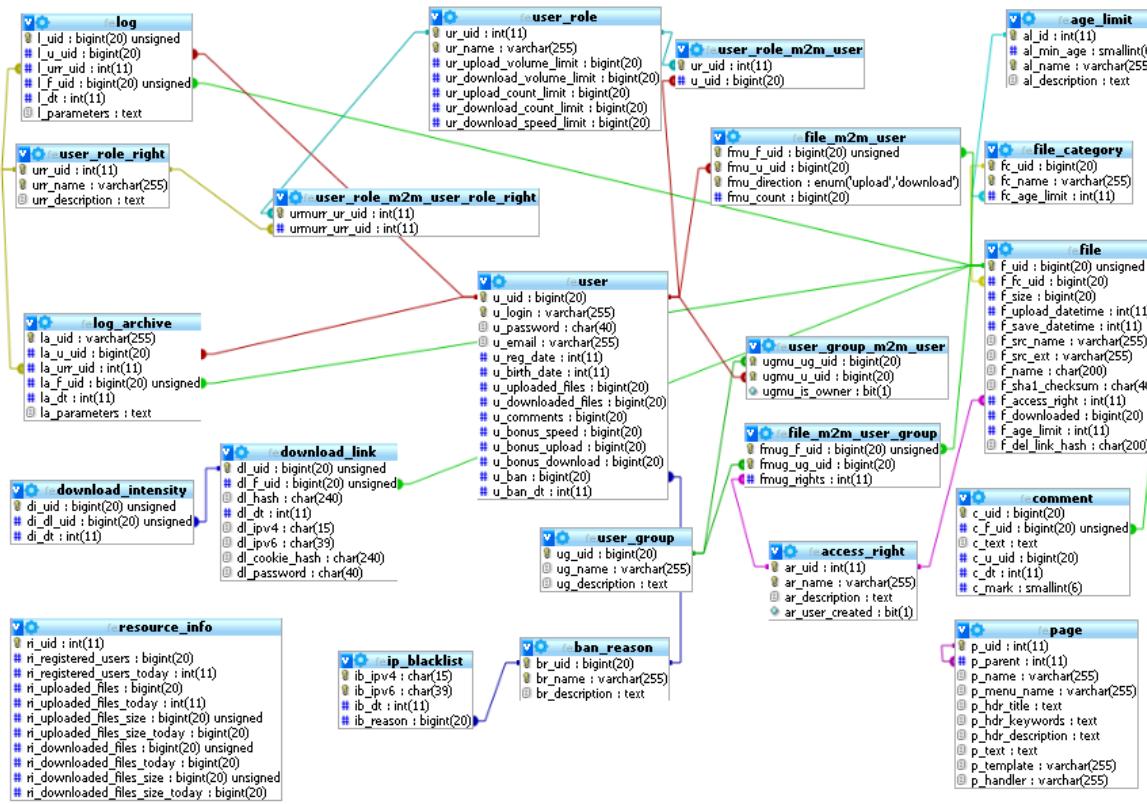
|   |   |  |
|---|---|--|
| Комментарии   | <pre> classDiagram     class comment {         &lt;&lt;columns&gt;&gt;         *PK c_uid: BIGINT         *FK c_f_uid: BIGINT UNSIGNED         • c_text: TEXT         • c_u_uid: BIGINT         • c_dt: INTEGER         c_mark SMALLINT          &lt;&lt;FKs&gt;&gt;         + FK_comment_file(BIGINT UNSIGNED)         &lt;&lt;PKs&gt;&gt;         + PK_comment(BIGINT)     }   </pre>  | <p>Плоская структура, т.е. нельзя комментировать комментарии.</p> <p>Оценку при комментировании можно не выставлять (NULL).</p> <p><i>Возможно, стоит сделать PK тоже UNSIGNED.</i></p>                  |
| Пользователи  | <pre> classDiagram     class user {         &lt;&lt;columns&gt;&gt;         *PK u_uid: BIGINT         • u_login: VARCHAR(255)         • u_password: CHAR(40)         • u_email: VARCHAR(255)         • u_reg_date: INTEGER         • u_birth_date: INTEGER         u_uploaded_files: BIGINT         u_downloaded_files: BIGINT         u_comments: BIGINT         u_bonus_speed: BIGINT         u_bonus_upload: BIGINT         u_bonus_download: BIGINT         FK u_ban: BIGINT         u_ban_dt: INTEGER          &lt;&lt;FKs&gt;&gt;         + FK_user_ban_reason(BIGINT)         &lt;&lt;PKs&gt;&gt;         + PK_user(BIGINT)         &lt;&lt;unique&gt;&gt;         + UQ_user_u_login(VARCHAR)     }   </pre> | Зарегистрированные пользователи файлообменного сервиса.  |
| Роли (пользователь может находиться в нескольких ролях) | <pre> classDiagram     class user_role {         &lt;&lt;columns&gt;&gt;         *PK ur_uid: INTEGER         ur_name: VARCHAR(255)         ur_upload_volume_limit: BIGINT         ur_download_volume_limit: BIGINT         ur_upload_count_limit: BIGINT         ur_download_count_limit: BIGINT         ur_download_speed_limit: BIGINT          &lt;&lt;PKs&gt;&gt;         + PK_user_role(INTEGER)         &lt;&lt;unique&gt;&gt;         + UQ_user_role_ur_name(VARCHAR)     }   </pre>   | Права ролей (правами обладают только роли, напрямую пользователям права давать нельзя)   |
| Файлы, закачанные или скачанные данным пользователем    | <pre> classDiagram     class file_m2m_user {         &lt;&lt;columns&gt;&gt;         *PK fmu_f_uid: BIGINT UNSIGNED         *PK fmu_u_uid: BIGINT         *PK fmu_direction: ENUM = {upload,down...}         fmu_count: BIGINT          &lt;&lt;FKs&gt;&gt;         + FK_file_m2o_user_file(BIGINT UNSIGNED)         + FK_file_m2o_user_user(BIGINT)         &lt;&lt;PKs&gt;&gt;         + PK_file_m2o_user(BIGINT UNSIGNED, BIGINT, ENUM)     }   </pre>   | Связь между пользователем и файлами. Здесь учитывается, закачал и/или скачал пользователь тот или иной файл (и сколько раз). В т.ч. учитывается, сколько раз пользователь закачивал файл с таким именем. |

|                          |   |   |
|--------------------------|---|---|
| Лог                      | <pre> log +-----+   *PK l_uid: BIGINT UNSIGNED   FK l_u_uid: BIGINT   *FK l_ur_uid: INTEGER   FK l_f_uid: BIGINT UNSIGNED     * l_dt: INTEGER     l_parameters: TEXT +-----+   *FKs   + FK_log_file(BIGINT UNSIGNED)   + FK_log_user(BIGINT)   + FK_log_user_role_right(INTEGER) +-----+   *PKs   + PK_log(BIGINT UNSIGNED)   </pre>                            | Хранит действия за последний месяц, потом они переносятся в архив лога.   |
| Архив лога               | <pre> log_archive +-----+   *PK la_uid: VARCHAR(255)   FK la_u_uid: BIGINT   *FK la_ur_uid: INTEGER   FK la_f_uid: BIGINT UNSIGNED     * la_dt: INTEGER     la_parameters: TEXT +-----+   *FKs   + FK_log_archive_file(BIGINT UNSIGNED)   + FK_log_archive_user(BIGINT)   + FK_log_archive_user_role_right(INTEGER) +-----+   *PKs   + PK_log(VARCHAR)   </pre> | <p>Здесь ПК является 255-разрядным 36-ричным числом. При перемещении записей из основного лога это число должно автоинкрементироваться. Продумать механизм переноса (возможно, стоит использовать хранимую процедуру).</p> <p><b>В эту таблицу НЕЛЬЗЯ писать данные «напрямую»!!! Можно только перемещать из таблицы log.</b></p> |
| Чёрный список IP-адресов | <pre> ip_blacklist +-----+   *PK ib_ipv4: CHAR(15)   *PK ib_ipv6: CHAR(39)     * ib_dt: INTEGER     FK ib_reason: BIGINT +-----+   *FKs   + FK_ip_blacklist_ban_reason(BIGINT) +-----+   *PKs   + PK_ip_blacklist(CHAR, CHAR)   </pre>  | Чёрный список ip-адресов (для бана зарегистрированных пользователей использовать таблицу пользователей).  |
| Причины блокировки       | <pre> ban_reason +-----+   *PK br_uid: BIGINT   br_name: VARCHAR(255)   br_description: TEXT +-----+   *PKs   + PK_ban_reason(BIGINT) +-----+   *unique   + UQ_ban_reason_br_name(VARCHAR)   </pre>   | Причины бана.   |

|                             |   |   |
|-----------------------------|---|---|
| Страницы сайта              | <pre> classDiagram     class page {         *PK p_uid: INTEGER         FK p_parent: INTEGER         * p_name: VARCHAR(255)         * p_menu_name: VARCHAR(255)         * p_hdr_title: TEXT         * p_hdr_keywords: TEXT         * p_hdr_description: TEXT         p_text: TEXT         * p_template: VARCHAR(255)         p_handler: VARCHAR(255)          &lt;&lt;FKs&gt;&gt;         + FK_page_page(INTEGER)          &lt;&lt;PKs&gt;&gt;         + PK_page(INTEGER)     }   </pre>   | p_parent – рекурсивный внешний ключ. Предусмотреть ограничение: p_parent может быть NULL только у ОДНОЙ (главной) страницы.   |
| Ссылки на скачивание файлов | <pre> classDiagram     class download_link {         *PK dl_uid: BIGINT UNSIGNED         *FK dl_f_uid: BIGINT UNSIGNED         * dl_hash: CHAR(240)         * dl_dt: INTEGER         dl_ipv4: CHAR(15)         dl_ipv6: CHAR(39)         * dl_cookie_hash: CHAR(240)         dl_password: CHAR(40)          &lt;&lt;FKs&gt;&gt;         + FK_download_link_file(BIGINT UNSIGNED)          &lt;&lt;PKs&gt;&gt;         + PK_download_link(BIGINT UNSIGNED)     }   </pre>  | После того, как пользователь закачал файл, он может скачать его из «личного кабинета» или «дать ссылку другу». Ссылка помещается в эту таблицу и имеет «срок годности». |
| Интенсивность скачиваний    | <pre> classDiagram     class download_intensity {         *PK di_uid: BIGINT UNSIGNED         *FK di_dl_uid: BIGINT UNSIGNED         di_dt: INTEGER          &lt;&lt;FKs&gt;&gt;         + FK_download_intensity_download_link(BIGINT UNSIGNED)          &lt;&lt;PKs&gt;&gt;         + PK_download_intensity(BIGINT UNSIGNED)     }   </pre>  | Итенсивность скачиваний (дополнение основного лога: сколько раз за некоторое время использовали конкретную ссылку для скачивания).                                      |
| Информация о ресурсе        | <pre> classDiagram     class resource_info {         *PK ri_uid: INTEGER         ri_registered_users: BIGINT         ri_registered_users_today: INTEGER         ri_uploaded_files: BIGINT         ri_uploaded_files_today: INTEGER         ri_uploaded_files_size: BIGINT UNSIGNED         ri_uploaded_files_size_today: BIGINT         ri_downloaded_files: BIGINT UNSIGNED         ri_downloaded_files_today: BIGINT         ri_downloaded_files_size: BIGINT UNSIGNED         ri_downloaded_files_size_today: BIGINT          &lt;&lt;PKs&gt;&gt;         + PK_resource_info(INTEGER)     }   </pre> | Все поля кэшируются на основе агрегирования данных из других таблиц.<br>В этой таблице должна быть ТОЛЬКО ОДНА запись.  |

|                      |  |  |
|----------------------|--|--|
| Группы пользователей |     | Группы пользователей (для создания "групп друзей", групп доступа к каким-то файлам и т.д.) Владелец группы отмечается в user_group_m2m_user. |
| <none>               |    | Связь между файлами и группами пользователей (с соответствующими правами).   |
| <none>               |    | Связь между пользователями и группами пользователей.   |
| <none>               |   | Связь между ролями пользователей и пользователями.   |
| <none>               |  | Связь между ролями пользователей и правами ролей пользователей.  |

Итак, схема БД получена. Её можно экспортить в СУБД и получить реально работающую БД.



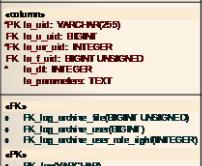
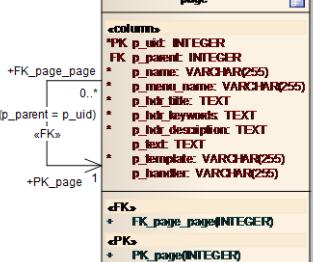
Сейчас необходимо ещё раз внимательно просмотреть всё, что у нас есть по даталогическому уровню, и добавить необходимые ограничения целостности и консистентности данных, реализуя соответствующие триггеры и хранимые подпрограммы (процедуры и функции).

Снова обратимся к таблице, в которой перечислены все таблицы БД, и отметим, где и какие императивные ограничения нужны.

**Примечание:** декларативными ограничениями являются *ключи и связи*; императивными (т.е. такими, которые «что-то делают») – *триггеры и хранимые подпрограммы*.

Здесь отразим только те таблицы БД, для которых необходимо добавить императивные ограничения.

| Таблица БД        | Императивные ограничения   | Комментарии  |
|-------------------|--|--|
|                   | f_name не должно совпадать с f_del_link_hash   | Одна из основных, наиболее часто используемых таблиц.<br><br><b>Максимально продумать оптимизацию производительности!</b><br><br>Файл сохраняется на сервере под именем, состоящим из пяти sha1-хешей, получаемых на основе имени файла и случайных величин. |
| <b>БЫЛО:</b><br>  | <b>На данном этапе анализа была выявлена ошибка проектирования. Неудобно и нелогично использовать ar_user_created, куда лучше – сделать ссылку на пользователя, создавшего право доступа.</b>  | Параметры доступа к файлу. Например, "публичный", "личный", "для друзей", "для указанного круга пользователей".  |
| <b>Стало:</b><br> | И также следует предусмотреть императивное ограничение: ar_name может совпадать для прав, созданных разными пользователями, но не может быть двух одноименных прав, созданных одним и тем же пользователем. Также имя права (ar_name), созданного пользователем, не может совпадать с именем «предустановленного права». | Пользователи могут создавать свои наборы прав доступа (ar_user_created == 1).<br><br>Изначально в таблице расположено несколько «предустановленных» прав, например: «файл виден всем», «файл виден только владельцу» и т.п.                                  |
| <b>Было:</b>      | Ещё одна ошибка проектирования: al_min_age должен быть беззнаковым. И, с точки зрения здравого смысла, не имеет делать категорию с физическими ограничениями до 10-ти лет (хотя,   | Возрастные ограничения. Наименование ограничения -- TBD, см. международную практику по маркировке фильмов, игр и   |

|   |   |   |
|---|---|---|
|  <p>Стало:</p>  | <p>надо бы уточнить у юристов).</p>   | <p>т.п.</p>   |
|   | <p>Дата регистрации должна быть больше даты рождения.</p> <p>Также только сейчас было обнаружено, что в этой таблице не хватает поля, которое бы хранило имя пользователя для отображения на страницах ресурса.</p>   | <p>Зарегистрированные пользователи файлообменного сервиса.</p>  |
|    | <p>Запретить прямую вставку в эту таблицу. Возможно, за счёт управления правами из хранимой процедуры. Реализовать перенос данных за последний (завершившийся к данному моменту) месяц из таблицы log.</p> <p>Сделать своё решение с автоинкрементом 255-разрядного 36-ричного ключа.</p> | <p>Здесь РК является 255-разрядным 36-ричным числом. При перемещении записей из основного лога это число должно автоинкрементироваться. Продумать механизм переноса (возможно, стоит использовать хранимую процедуру).</p> <p><b>В эту таблицу НЕЛЬЗЯ писать данные «напрямую»!!! Можно только перемещать из таблицы log.</b></p> |
|    | <p>p_parent может быть NULL только у одной страницы (главной страницы сайта).</p>   | <p>p_parent – рекурсивный внешний ключ. Предусмотреть ограничение: p_parent может быть NULL только у ОДНОЙ (главной) страницы.</p>  |

|  |  |   |
|--|--|---|
|  | <p>Потребность наполнять эту табличу данными приводит к необходимости написать «агgregирующие хранимые процедуры», которые будут вычитывать данные из других таблиц и записывать сюда. Вешать отдельные триггеры на другие таблицы нерационально – это сильно скажется на производительности.</p> <p>Также имеет смысл добавить сюда поле с информацией о том, когда эта кэширующая таблица последний раз обновлялась.</p> | <p>Все поля кэшируются на основе агрегирования данных из других таблиц.</p> <p>В этой таблице должна быть ТОЛЬКО ОДНА запись.</p> |
|  |  |   |

Итак, на текущий момент мы в общих чертах продумали императивные ограничения. Практика показывает, что в процессе реализации приложения будут выявлены иные императивные и декларативные ограничения, которые сейчас просто неочевидны. Также более вдумчивый анализ модели БД позволил выявить в ней несколько ошибок и прийти к выводу о том, что везде, где мы не собираемся хранить отрицательный числа, а само хранимое значение может быть большим, имеет смысл использовать BIGINT UNSIGNED.

Приступим к написанию триггеров и хранимых подпрограмм.

| Таблица БД | Императивные ограничения                            | Код   |
|------------|---|---|
|            | <p>f_name не должно совпадать с f_del_link_hash</p> | <pre> DROP TRIGGER IF EXISTS `unique_hashes_bi`; delimiter // CREATE TRIGGER `unique_hashes_bi` BEFORE INSERT ON `file` FOR EACH ROW BEGIN     IF NEW.`f_name` = NEW.`f_del_link_hash` THEN         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Hashes for file name and deletion link can not be equal', MYSQL_ERRNO = 1001;     END IF; END // delimiter ;  DROP TRIGGER IF EXISTS </pre> |

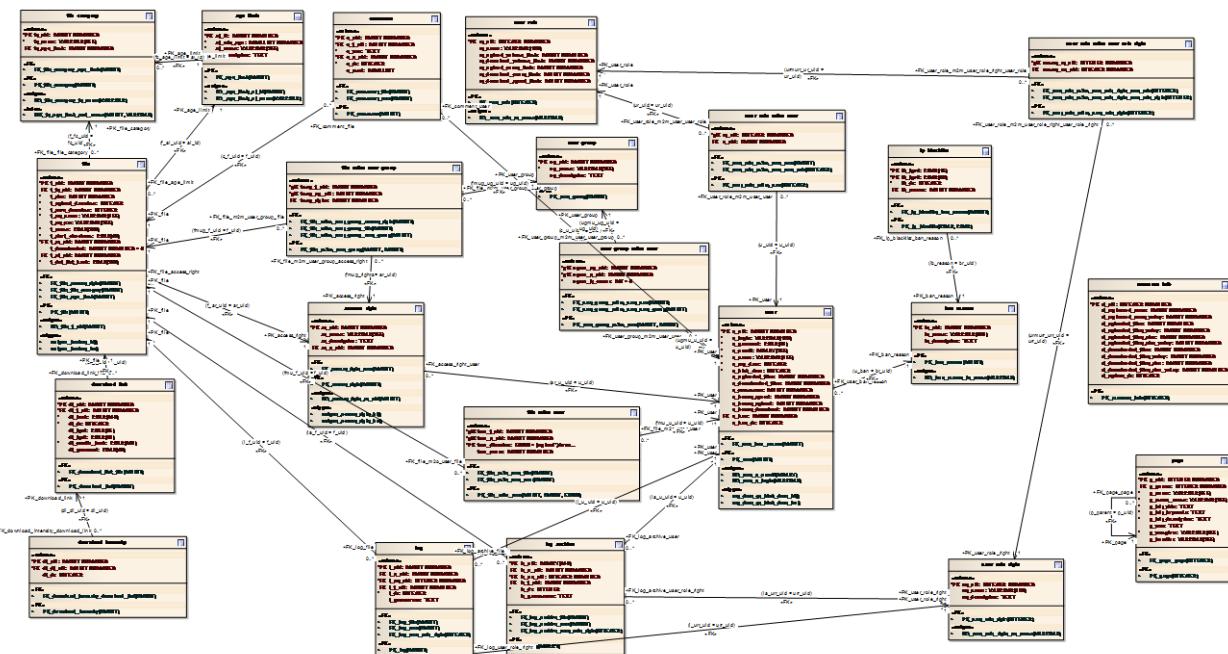
|  |  |  |
|--|--|--|
|  |  | <pre> `unique_hashes_bu`;  delimiter //  CREATE TRIGGER `unique_hashes_bu` BEFORE UPDATE ON `file` FOR EACH ROW BEGIN     IF NEW.`f_name` = NEW.`f_del_link_hash` THEN         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Hashes for file name and deletion link can not be equal', MYSQL_ERRNO = 1001;     END IF; END //    delimiter ; </pre>  |
|  |  | <p>ar_name может совпадать для прав, созданных разными пользователями, но не может быть двух одноимённых прав, созданных одним и тем же пользователем. Также имя права (ar_name), созданного пользователем, не может совпадать с именем «предустановленного права».</p> <pre> DROP TRIGGER IF EXISTS `unique_access_right_bi`;  delimiter //  CREATE TRIGGER `unique_access_right_bi` BEFORE INSERT ON `access_right` FOR EACH ROW BEGIN     DECLARE duplicates integer;      SELECT count(*) INTO duplicates     from `access_right` where     `ar_name` = NEW.`ar_name` AND     (ISNULL(`ar_u_uid`) OR     `ar_u_uid` = NEW.`ar_u_uid`);      IF duplicates &gt; 0 THEN         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Access right name must be unique for one user and must not be similar to predefined name', MYSQL_ERRNO = 1001;     END IF;  END //    delimiter ;  DROP TRIGGER IF EXISTS `unique_access_right_bu`;  delimiter //  CREATE TRIGGER `unique_access_right_bu` BEFORE UPDATE ON `access_right` FOR EACH ROW </pre> |

|  |   |  |
|--|---|--|
|  |   | <pre> BEGIN DECLARE duplicates integer;  SELECT count(*) INTO duplicates from `access_right` where `ar_name`=NEW.`ar_name` AND (ISNULL(`ar_u_uid`) OR `ar_u_uid`=NEW.`ar_u_uid`);  IF duplicates &gt; 0 THEN     SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Access right name must be unique for one user and must not be similar to predefined name', MYSQL_ERRNO = 1001; END IF;  END //</pre> <p>delimiter ;</p>   |
|  |    | <p>Не имеет смысла делать категорию с возрастными ограничениями до 10-ти лет.</p> <p><b>Задание для самоподготовки:</b> создать соответствующие ограничения самостоятельно.</p>  |
|  |  | <p>Дата регистрации должна быть больше даты рождения.</p> <pre> DROP TRIGGER IF EXISTS `reg_date_gt_birth_date_bi`;  delimiter //  CREATE TRIGGER `reg_date_gt_birth_date_bi` BEFORE INSERT ON `user` FOR EACH ROW BEGIN     IF NEW.`u_reg_date` &lt;= NEW.`u_birth_date` THEN         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Registration date can not be less or equal to birth date', MYSQL_ERRNO = 1001;     END IF; END //</pre> <p>delimiter ;</p> <pre> DROP TRIGGER IF EXISTS `reg_date_gt_birth_date_bu`;  delimiter //  CREATE TRIGGER `reg_date_gt_birth_date_bu` BEFORE UPDATE ON `user` FOR EACH ROW BEGIN     IF NEW.`u_reg_date` &lt;= NEW.`u_birth_date` THEN</pre> |

|  |  |  |
|--|--|--|
|  |  | <pre> SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Registration date can not be less or equal to birth date', MYSQL_ERRNO = 1001; END IF; END //</pre> <p>delimiter ;</p> |
|  | <p>Запретить прямую вставку в эту таблицу. Возможно, за счёт управления правами из хранимой процедуры. Реализовать перенос данных за последний (завершившийся к данному моменту) месяц из таблицы log.</p> <p><b>Сделать своё решение с автоинкрементом 255-разрядного 36-ричного ключа.</b></p> <p>Анализ показал, что эффективнее использовать шесть sha1 хэшей: от исходного PK, времени и 4-х случайных чисел. В схему внесены изменения – поле la_uid изменено на char(240)</p> | <b>Задание для самоподготовки:</b> реализовать решение самостоятельно.   |
|  | <p>p_parent может быть NULL только у одной страницы (главной страницы сайта).</p>  | <b>Задание для самоподготовки:</b> создать соответствующие ограничения самостоятельно.   |
|  | <p>dl_hash и dl_cookie_hash не должны совпадать.</p>   | <b>Задание для самоподготовки:</b> создать соответствующие ограничения самостоятельно.   |

|  |  |   |
|--|--|---|
| <pre> resource_info +-----+  PK  i_id: INTEGER       i_registered_user: BIGINT       i_registered_user_today: INTEGER       i_uploaded_file: BIGINT       i_uploaded_file_today: INTEGER       i_uploaded_file_size: BIGINT UNSIGNED       i_uploaded_file_size_today: BIGINT        PKs  + PK_resource_info( INTEGER ) </pre> | <p>Потребность наполнять эту таблицу данными приводит к необходимости написать «агрегирующие хранимые процедуры», которые будут вычитывать данные из других таблиц и записывать сюда. Вешать отдельные триггеры на другие таблицы нерационально – это сильно скажется на производительности.</p> | <p><b>Задание для самоподготовки:</b> создать соответствующие ограничения самостоятельно.</p> |
|--|--|---|

Итак, после реализации императивных ограничений и исправления всех обнаруженных недоработок схема БД приняла следующий вид:



На данном этапе только лишь графического представления уже недостаточно – нужно иметь возможность просматривать свойства тех или иных объектов, таким образом с «распечаткой» работать будет очень тяжело – нужно полностью переключаться на работу со средством автоматизированного проектирования.

## 4.4. Проектирование БД на физическом уровне

Вспомним определение физического уровня моделирования.

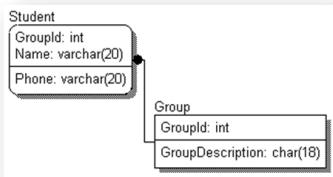
### Физический уровень

Таблицы, связи, индексы, методы хранения, настройки производительности, безопасности и т.п.

#### Описывать модель здесь можно

«Как угодно» (нет общепринятых решений)

Чаще всего выбирают такой же способ представления, как и у даталогической модели.



Из-за разнообразия предметных областей здесь используется всё – от чертежей и схем, до специальных нотаций (IDEF0).

На данном уровне необходимо продумать и реализовать следующие параметры БД.

#### Права доступа

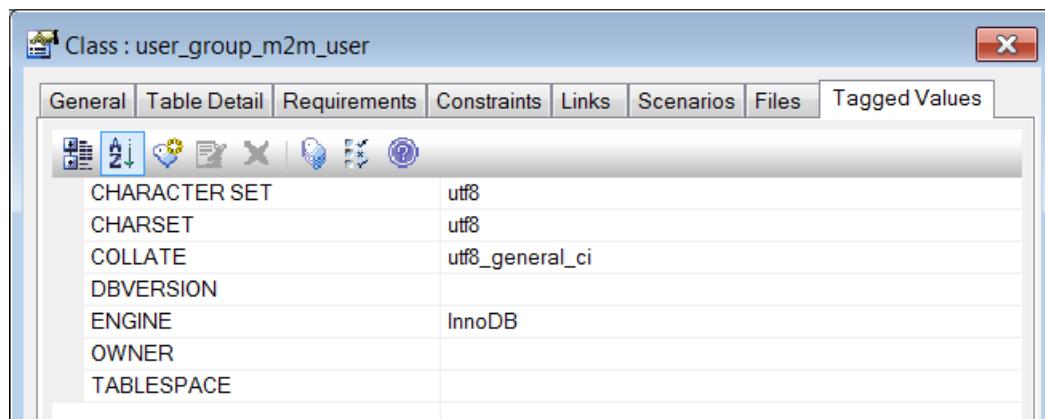
Права доступа управляются непосредственно самой СУБД, но к проектируемой модели БД можно приложить готовые скрипты по созданию пользователей и передаче им соответствующих прав на те или иные объекты БД.

**Задание для самоподготовки:** продумать вопрос распределения прав, написать соответствующие скрипты.

#### Кодировки (в т.ч. по умолчанию)

Хороший скрипт создания БД должен управлять кодировками явно, но если это невозможно или затруднительно по той или иной причине, можно перенести управление кодировками на прилагаемые к модели БД дополнительные скрипты или указать соответствующие пункты в инструкции по настройке MySQL.

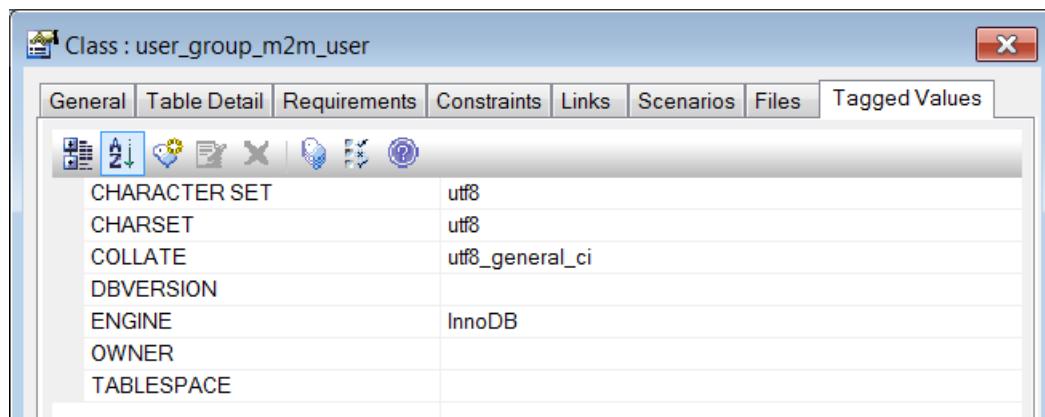
Поскольку мы используем Sparx EA, мы можем указать кодировки через свойство Tagged Values таблицы (создав свои дополнительные значения, см. подробнее [здесь](#)).



В процессе указания кодировок выяснилось, что даже MySQL 5.5.x не может использовать первичные ключи в UTF8 длиннее 180-200 символов (в пересчёте на байты получается превышение максимального размера индекса), что привело к необходимости исправления модели: в некотором случае длина текстовых первичных ключей была уменьшена до 180 символов (вместо 255), в некоторых случаях был изменён тип данных (BINARY вместо VARCHAR, т.к. BINARY – это «просто байты, без кодировки»).

### Методы доступа и параметры хранения файлов таблиц

Многие универсальные средства проектирования не позволяют проводить столь тонкую настройку СУБД (и тогда придётся решать этот вопрос дополнительными скриптами или инструкцией по настройке СУБД), но в Sparx EA это делается, опять же, через Tagged Values таблицы см. подробнее [здесь](#)).



## Индексы

Поскольку индексы (за исключением первичных ключей и ограничений уникальности) не влияют на логику работы БД, создание индексов лучше отложить до момента наполнения БД тестовыми данными и проведения нагружочного тестирования. Однако, самые очевидные индексы можно создать уже сейчас, тем более, что Sparx EA позволяет это сделать.

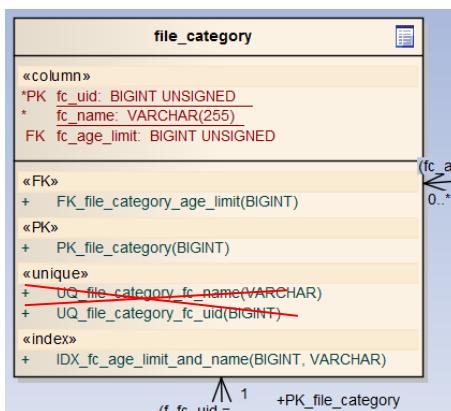
Будьте внимательны: в некоторых случаях создание индекса в Sparx EA приводит с «сбрасыванием» сведений о внешних ключах в связях. Если это произошло (при импорте модели в БД вы получаете ошибку № 150), просто выполните заново операцию установки внешних ключей между таблицами, при создании которых возникала ошибка № 150.

Ещё одно замечание по индексам: Sparx EA часто генерирует ОТДЕЛЬНО команды по созданию первичных ключей и уникальных индексов на тех же полях, которые входят в состав первичных ключей. Это легко заметить по сообщениям, например, phpMyAdmin:

| Action     | Keyname                   | Type  | Unique | Packed | Column       | Cardinality | Collation | Null | Comment |
|------------|---------------------------|-------|--------|--------|--------------|-------------|-----------|------|---------|
| Edit  Drop | PRIMARY                   | BTREE | Yes    | No     | fc_uid       | 0           | A         |      |         |
| Edit  Drop | UQ_file_category_fc_name  | BTREE | Yes    | No     | fc_name      | 0           | A         |      |         |
| Edit  Drop | UQ_file_category_fc_uid   | BTREE | Yes    | No     | fc_uid       | 0           | A         |      |         |
| Edit  Drop | fc_age_limit              | BTREE | No     | No     | fc_age_limit | 0           | A         | YES  |         |
| Edit  Drop | IDX_fc_age_limit_and_name | BTREE | No     | No     | fc_age_limit | 0           | A         | YES  |         |
|            |                           |       |        |        | fc_name      | 0           | A         |      |         |

The indexes PRIMARY and UQ\_file\_category\_fc\_uid seem to be equal and one of them could possibly be removed.

В таком случае имеет смысл удалить в модели БД операцию, отвечающую за уникальность первичного ключа (СУБД и так понимает, что он должен быть уникальным).



И теперь всё хорошо:

| Action     | Keyname                   | Type  | Unique | Packed | Column       | Cardinality | Collation | Null | Comment |
|------------|---------------------------|-------|--------|--------|--------------|-------------|-----------|------|---------|
| Edit  Drop | PRIMARY                   | BTREE | Yes    | No     | fc_uid       | 0           | A         |      |         |
| Edit  Drop | UQ_file_category_fc_name  | BTREE | Yes    | No     | fc_name      | 0           | A         |      |         |
| Edit  Drop | fc_age_limit              | BTREE | No     | No     | fc_age_limit | 0           | A         | YES  |         |
| Edit  Drop | IDX_fc_age_limit_and_name | BTREE | No     | No     | fc_age_limit | 0           | A         | YES  |         |
|            |                           |       |        |        | fc_name      | 0           | A         |      |         |

**Задание для самоподготовки:** продумать, создать и прокомментировать индексы для остальных таблиц.

## Настройки СУБД

Пожалуй, это самые сложные параметры для настройки через внешние средства проектирования. Если для корректной и эффективной работы БД необходимо настроить СУБД некоторым особым образом, необходимо исследовать возможности СУБД по управлению такими настройками через SQL и написать соответствующие скрипты, однако часть настроек, вероятно, придётся корректировать средствами администрирования СУБД.

В любом случае – эти настройки должны быть подробно задокументированы.

**Задание для самоподготовки:** продумать и описать, какие настройки MySQL 5.5.x могут повлиять на показатели качества проектируемой БД, и какие их значения необходимо выставить для достижения наилучшей работоспособности БД.

## Итог

Итак, процесс проектирования завершён. После того, как вы выполните задания и доведёте построение модели до логического завершения, останется её только протестировать.

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

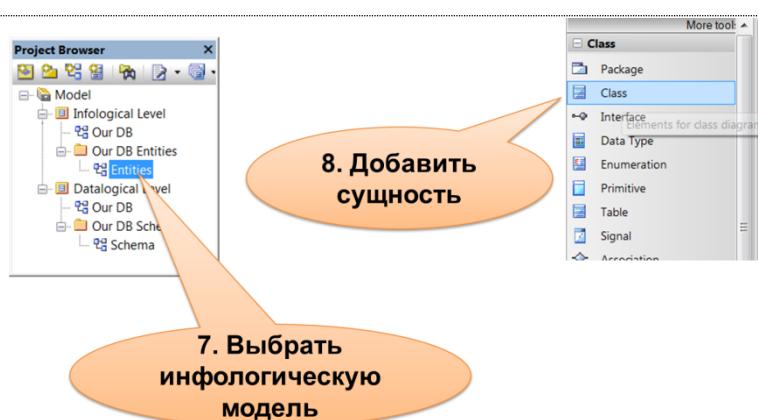
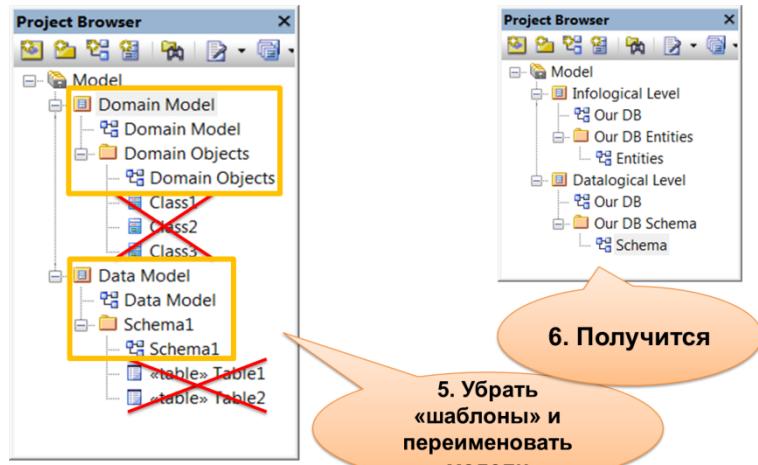
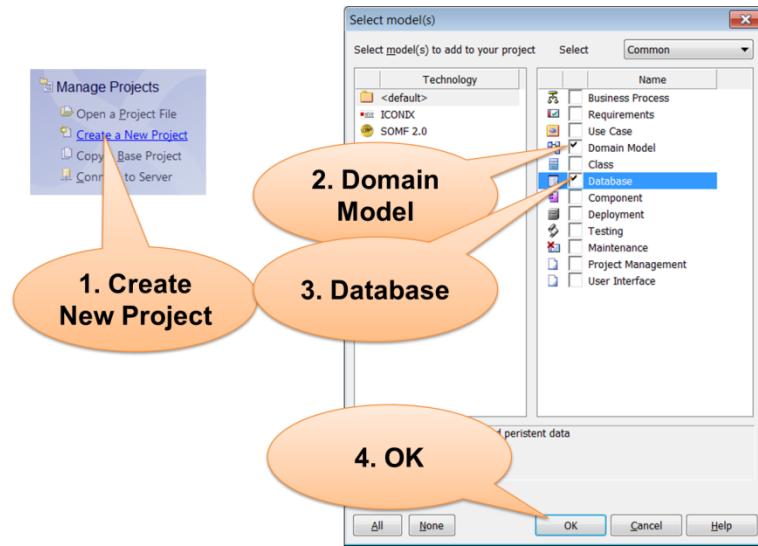
---

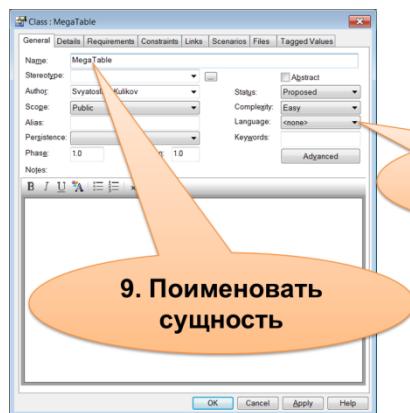
---

## 4.5. Основы использования Sparx EA

Disclaimer: данный раздел представляет собой просто набор картинок, в которых по шагам описаны самые основные действия. За более подробной информацией обращайтесь к официальной документации Sparx EA.

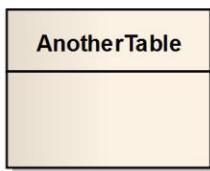
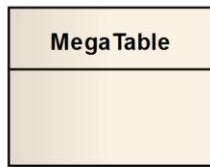
### 4.5.1. Создание модели и работа с инфологическим уровнем



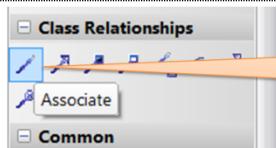


10. Язык реализации = <none>

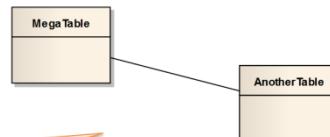
9. Поименовать сущность



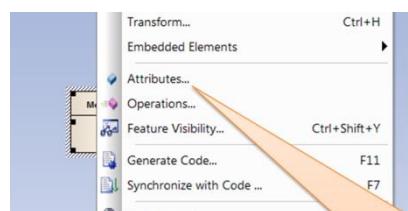
11. Создать так все необходимые сущности



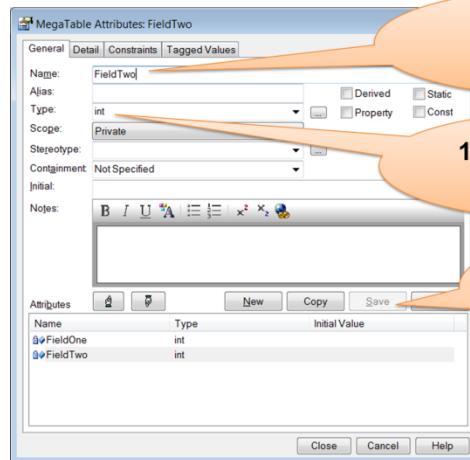
12. Associate



13. Провести так все необходимые связи



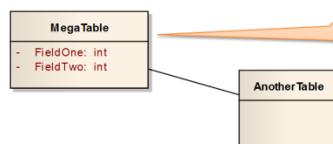
**14. Правой кнопкой мыши вызвать меню работы с сущностью, в нём выбрать Attributes**



**15. Указать имя атрибута**

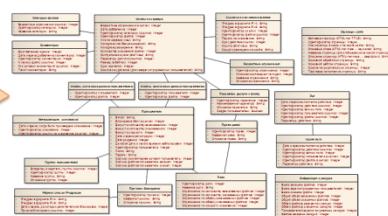
**16. Указать общий тип данных**

**17. Сохранить**

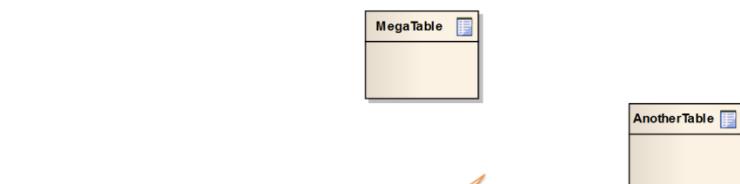
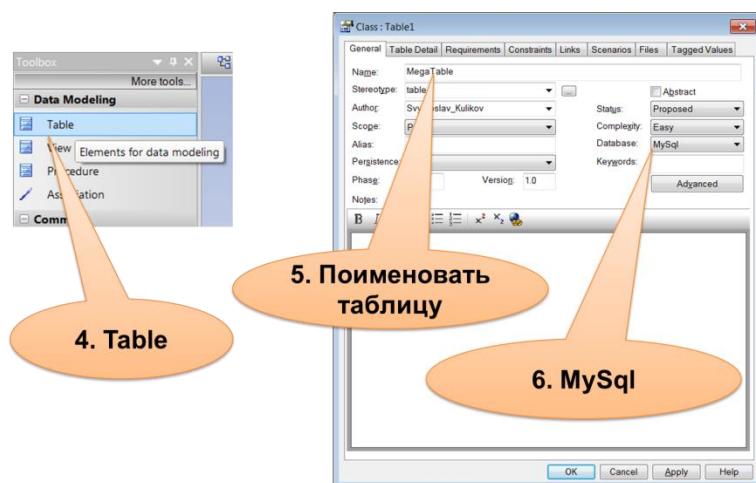
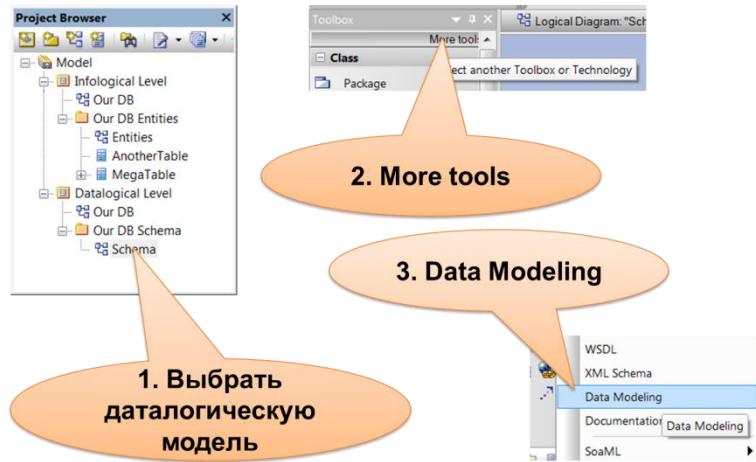


**18. Повторять...**

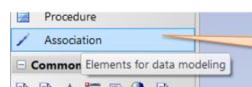
**19. ... до заполнения атрибутами всех сущностей**



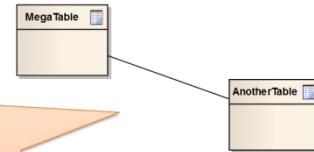
## 4.5.2. Работа с даталогическим и физическим уровнем



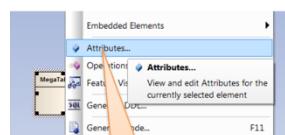
**7. Создать так все необходимые таблицы**



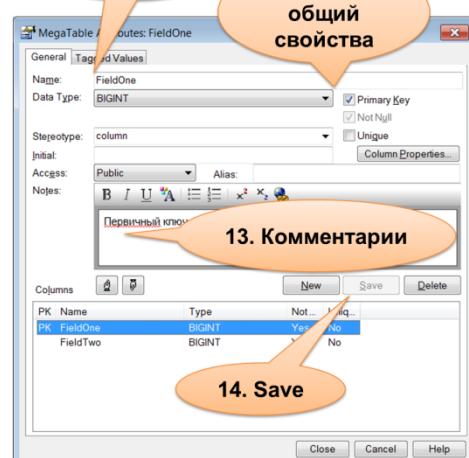
**8. Association**



**9. Связи проводятся от ДОЧЕРНЕЙ таблицы к РОДИТЕЛЬСКОЙ**



**10. Attributes**

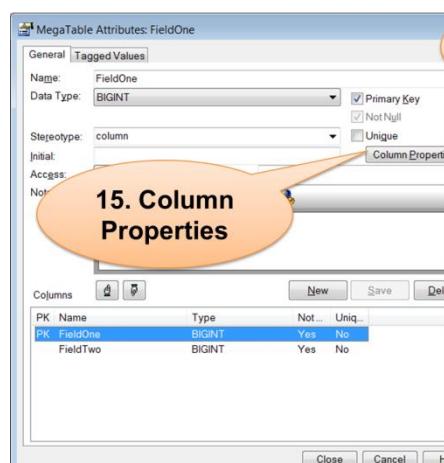


**11. Имя**

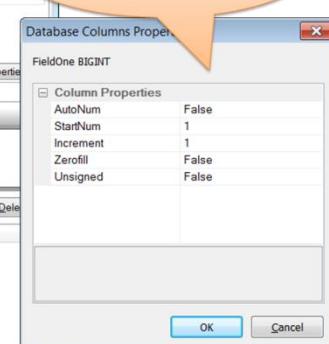
**12. Тип и общий свойства**

**13. Комментарии**

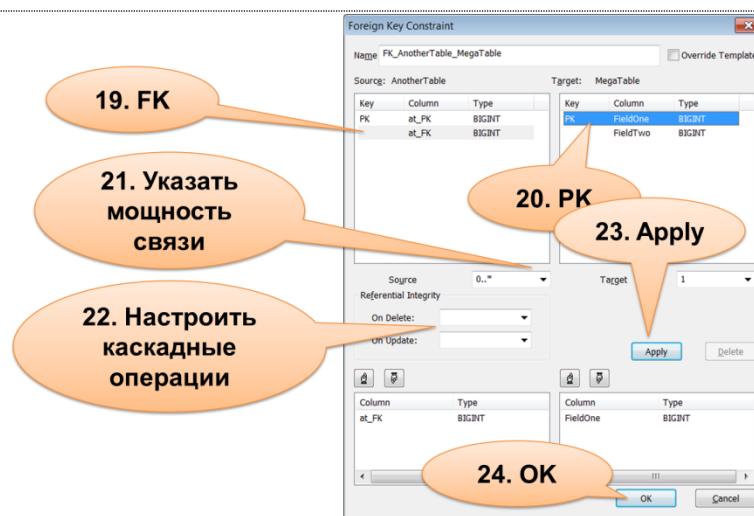
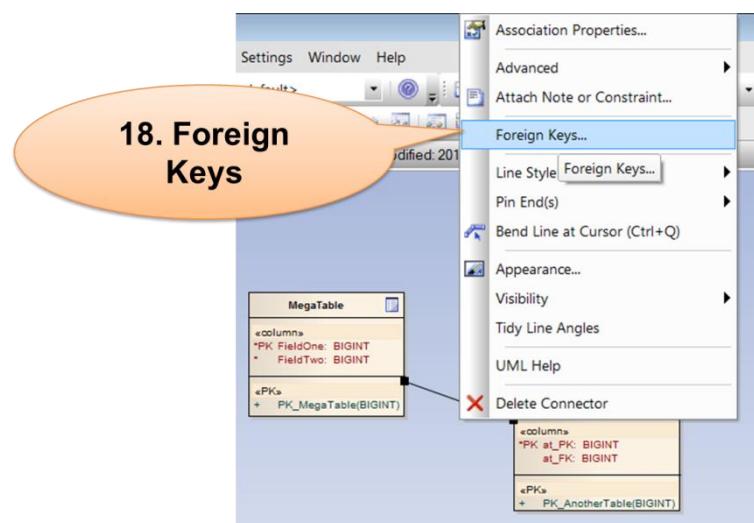
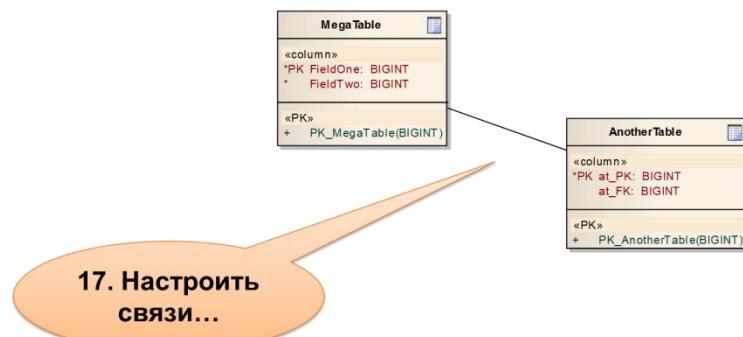
**14. Save**

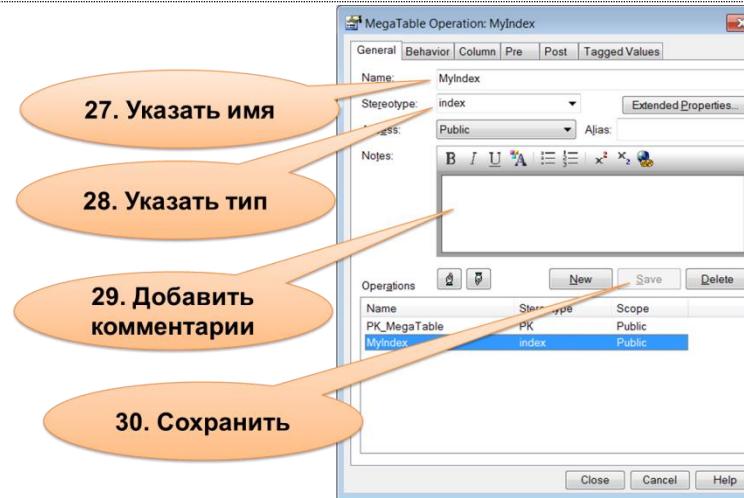
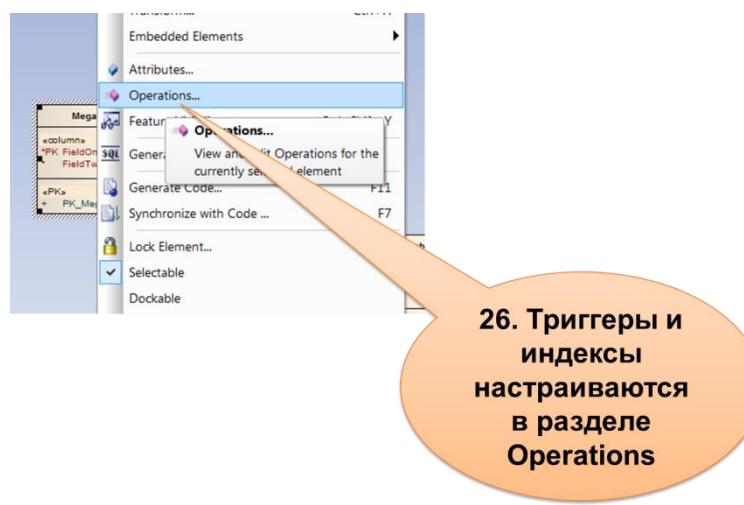
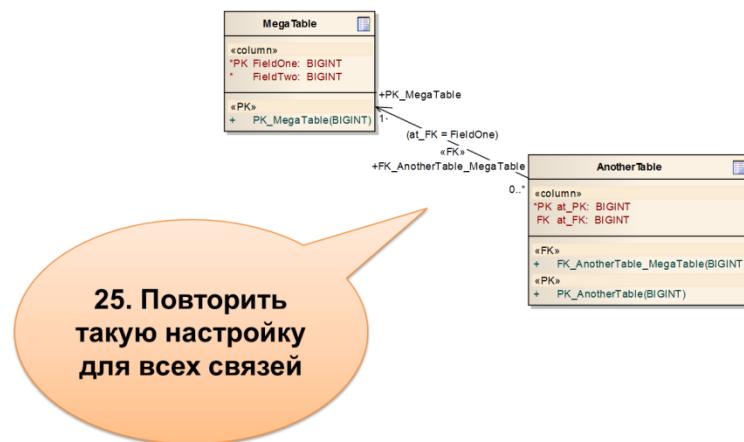


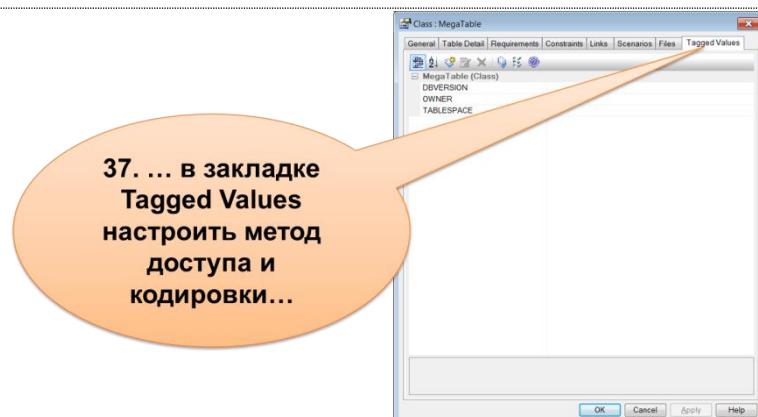
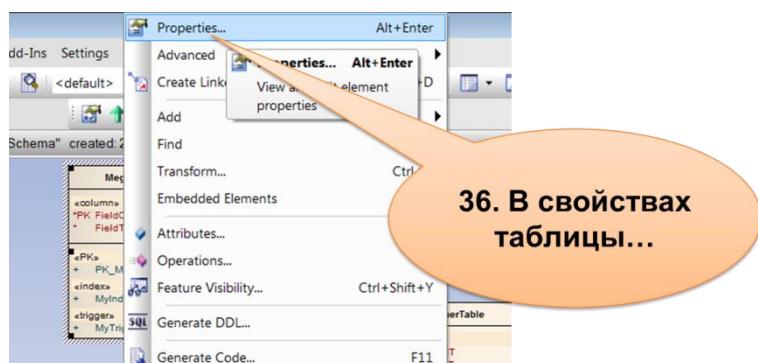
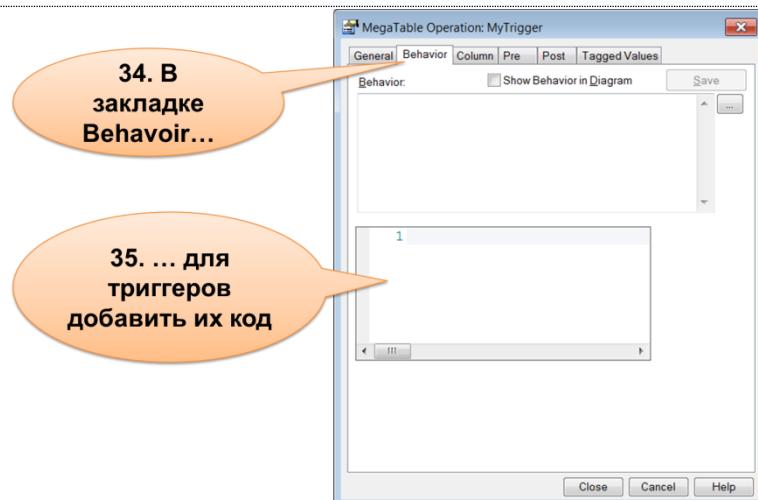
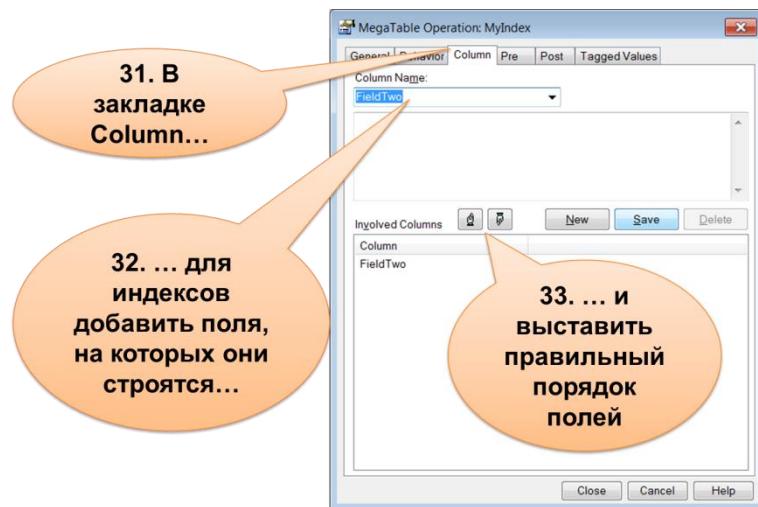
**15. Column Properties**

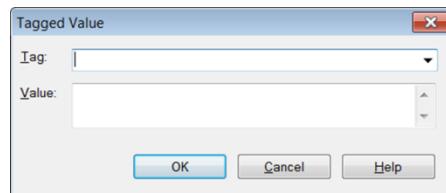


**16. Настроить нужное**



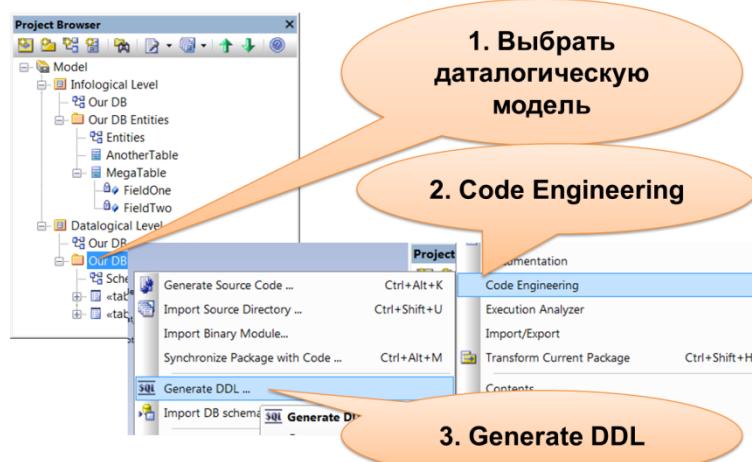




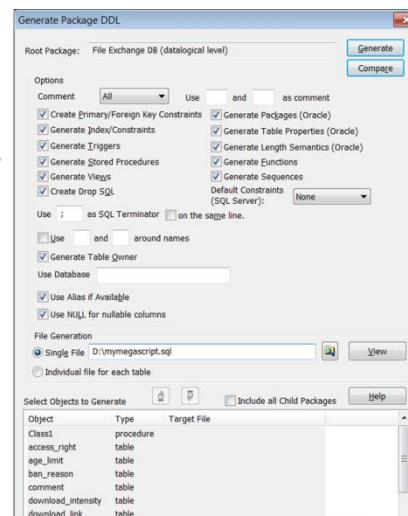


**38. ... путём  
добавления новых  
Tagged Values.**

#### 4.5.3. Создание скрипта генерации БД



**4. Настроить параметры генерации DDL и выполнить генерацию**



```
CREATE TABLE access_rights
(
    ar_id INT NOT NULL AUTO_INCREMENT COMMENT 'Идентификатор параметра.',
    ar_name VARCHAR(100) NOT NULL COMMENT 'Наименование параметра (и т.д. может находиться пользовательским синонимом), когда вставляется флаг ar_user_created=1',
    ar_desc VARCHAR(100) NOT NULL COMMENT 'Описание параметра.',
    ar_value VARCHAR(100) NOT NULL COMMENT 'Значение параметра. NULL == предустановленное право.',
    PRIMARY KEY (ar_id),
    UNIQUE INDEX ar_id_idx,
    KEY ar_id_idx,
    INDEX ar_name_idx,
    INDEX ar_desc_idx
) DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci ENGINE=InnoDB COMMENT='Параметры доступа к файлу. Например, "зубчанин", "песни", "зап дружд", "для указанного круга поиска';

CREATE TABLE app_links
(
    al_id INT NOT NULL AUTO_INCREMENT COMMENT 'Идентификатор ограничения.',
    al_min_qty SMALLINT NOT NULL COMMENT 'Минимальный квота (%)', на который user не распространяется ограничение (например: 10 -- значит, что ограничение распространяется на тех, кого от него меньше 10%)
    al_max_qty SMALLINT NOT NULL COMMENT 'Наимаксимум ограничения -- TSL, с. максимальные ограничения по категориям фильма, игр и т.д. Не может быть 2^4 = 16 (единицами ограничений).',
    al_description TEXT NOT NULL COMMENT 'Описание ограничения (например: "Смотреть элементы некой категории, не для детей до 18 лет")',
    PRIMARY KEY (al_id),
    UNIQUE INDEX al_id_idx,
    UNIQUE INDEX al_max_qty_idx,
    INDEX al_min_qty_idx,
    INDEX al_desc_idx
) DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci ENGINE=InnoDB COMMENT='Ограничение ограничения. Наимаксимум ограничения -- TSL, с. максимальную цену по макоровки фил
```

**5. Полученный SQL-код импортировать в БД средствами СУБД или утилит администрирования БД/СУБД**

## Тема 5: Основы языка SQL (на базе MySQL)

### 5.1. Общие сведения об SQL

SQL (Structured Query Language, язык структурированных запросов) – универсальный язык, применяемый в реляционных СУБД для выполнения следующих классов операций.

| Класс операций            | Пример запроса   |
|---------------------------|--|
| Администрирование БД/СУБД | CREATE USER 'user'@'localhost'<br>IDENTIFIED BY 'password'                     |
| Управление структурой БД  | DROP TABLE IF EXISTS<br>'some table'   |
| Управление данными        | SELECT `field_one`, `field_two`<br>FROM `some_table` ORDER BY<br>'field three' |

Язык SQL основан на запросах (query) – инструкциях, оформленных определённым образом.

Каждая такая инструкция «просит» СУБД выполнить определённое действие, например извлечь данные, создать таблицу и т.п.

Все инструкции SQL имеют одинаковую структуру, которая изображена на рисунке.



## 5.2. Именование структур в MySQL

### Правила именования структур: простой вариант для запоминания

Следуя этим простым правилам, вы избавите себя от многих проблем:

- ТОЛЬКО английский язык.
- Допустимы буквы, цифры, знак подчёркивания.
- Не более 64 символов (16 для меток).
- Все идентификаторы заключены в обратные кавычки (` `).
- Все строковые константы заключены в апострофы (' ').

Пример правильно оформленного запроса

```
SELECT * FROM `t_1` WHERE `t_1`.`id`>100
```

Поскольку полноценное описание всех правил именования структур MySQL занимает достаточно большой объём, рекомендуется ознакомиться с ним здесь: <http://dev.mysql.com/doc/refman/5.5/en/identifiers.html>.

Вы всегда можете указать здесь последнюю версию MySQL или ту версию, с которой вы сейчас работаете.

### Чувствительность имён к регистру

SQL – нерегистрочувствительный язык, однако во многих (часто непредвиденных) случаях проблемы могут возникнуть с регистрационностью имён структур.

Подробности см. здесь: <http://dev.mysql.com/doc/refman/5.5/en/identifier-case-sensitivity.html>

А для повседневной жизни стоит запомнить простое правило: **«будем считать, что имена структур регистрочувствительны, и поступать соответственно».**

### Комментарии в MySQL

MySQL поддерживает следующие способы задания комментариев:

- с помощью символа '#', за которым следует текст комментария до конца строки;
- с помощью двух символов --, за которыми идёт текст комментария до конца строки;
- и (для многострочных комментариев) с помощью символов /\* (начало комментария) и \*/ (конец комментария).

Обратите внимание: при использовании для комментирования способа с -- (двойное тире) требуется наличие хотя бы одного пробела после второго тире!

## Пример использования комментариев

```
-- MySQL dump 10.13 Distrib 5.5.13, for Win32 (x86)
-- Host: localhost      Database: dml
-----
-- Server version 5.5.13

--
-- Table structure for table `payment`
--

DROP TABLE IF EXISTS `payment`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `payment` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) NOT NULL,
  `money` int(11) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;
```

## 5.3. Типы данных в MySQL

### Общие сведения о строках в MySQL

Строка – последовательность символов, заключенных либо в одинарные кавычки (' '), либо в двойные кавычки (" "). При использовании диалекта ANSI SQL допустимы только одинарные кавычки.

Например: 'a string'.

Внутри строки некоторые последовательности символов имеют специальное назначение, каждая из этих последовательностей начинается обратным слешем (\):

```
\0 \' \" \b \n \r \t \z \\ \% \_
```

Подробнее о строках в MySQL см. здесь:

<http://dev.mysql.com/doc/refman/5.5/en/string-literals.html>

### Кавычки внутри строк MySQL

Существует несколько способов включить кавычки в строку:

```
SELECT 'hello', '"hello"', """hello""", 'hel''lo', '\'hello'
      hello    "hello"    ""hello""    hel'lo      'hello
SELECT "hello", "'hello'", '''hello''', "hel""lo", "\"hello"
      hello    'hello'    ''hello''   hel"lo      "hello"
```

Универсальное решение: строка берётся в апострофы, внутренние апострофы экранируются, это будет выглядеть так:

```
SELECT 'Using a \'very\' "special" string'
      Using a 'very' "special" string
```

### Общие сведения о числах в MySQL

Целые числа представляются в виде последовательности цифр.

Для чисел с плавающей точкой в качестве разделителя десятичных знаков используется символ '.' (точка).

Числа обоих типов могут предваряться символом '-' (минус).

Примеры допустимых чисел:

```
1221 0 -32 294.42 -32032.6809e+10
```

## Числовые типы данных

| Тип                           | Байтов     | Минимальное значение                      | Максимальное значение                     |
|-------------------------------|------------|---|---|
|                               |            | (Знаковый/Беззнаковый)                    | (Знаковый/Беззнаковый)                    |
| TINYINT                       | 1          | -128                                      | 127                                       |
|                               |            | 0   | 255                                       |
| SMALLINT                      | 2          | -32768                                    | 32767                                     |
|                               |            | 0   | 65535                                     |
| MEDIUMINT                     | 3          | -8388608                                  | 8388607                                   |
|                               |            | 0   | 16777215                                  |
| INT                           | 4          | -2147483648                               | 2147483647                                |
|                               |            | 0   | 4294967295                                |
| BIGINT                        | 8          | -9223372036854775808                      | 9223372036854775807                       |
|                               |            | 0   | 18446744073709551615                      |
| DECIMAL<br>NUMERIC            | 65<br>symb | - 64 '9' (зависит от точно-<br>сти)       | 65 '9' (зависит от точно-<br>сти)         |
| FLOAT                         | 4          | (зависит от точности, до<br>23 зн. п. з.) | (зависит от точности, до<br>23 зн. п. з.) |
| DOUBLE<br>(PRECISION)<br>REAL | 8          | (зависит от точности, до<br>53 зн. п. з.) | (зависит от точности, до<br>53 зн. п. з.) |
| BIT                           | 64 bit     | 0   | 64 bits '1'                               |

## Типы данных даты и времени

|             |   |
|-------------|---|
| Тип столбца | Формат и запись «значения ноль»                                 |
| DATETIME    | «0000-00-00 00:00:00»   |
| DATE        | «0000-00-00»  |
| TIMESTAMP   | 00000000000000 (длина зависит от количества выводимых символов) |
| TIME        | «00:00:00»  |
| YEAR        | 00  |

Для типов DATETIME, DATE, TIMESTAMP и YEAR даты с неоднозначным годом интерпретируются в MySQL по следующим правилам:

- 00-69 конвертируется в 2000-2069;
- 70-99 конвертируется в 1970-1999.

## Строковые типы данных: CHAR и VARCHAR

Типы данных CHAR и VARCHAR различаются по способам хранения и извлечения.

Столбцы типа **CHAR**:

- хранят строки фиксированной длины (0-255);
- строки при хранении дополняются справа пробелами до заданной длины (эти пробелы удаляются при извлечении данных).

Столбцы **VARCHAR**:

- хранят строки переменной длины (0-65535);
- реальный размер при хранении равен ДЛИНА + 2 байта;
- хранимые данные пробелами не дополняются, наоборот, концевые пробелы при хранении удаляются.

## Строковые типы данных: BLOB и TEXT

Эти типы предназначены для хранения текстов и бинарных данных.

Модификации **BLOB** (регистрочувствительный тип):

- TINYBLOB (2<sup>8</sup>);
- BLOB (2<sup>16</sup>);
- MEDIUMBLOB (2<sup>24</sup>);
- LONGBLOB (2<sup>32</sup>).

Модификации **TEXT** (нерегистрочувствительный тип):

- TINYTEXT (2<sup>8</sup>);
- TEXT (2<sup>16</sup>);
- MEDIUMTEXT (2<sup>24</sup>);
- LONGTEXT (2<sup>32</sup>).

## Множества и перечисления

**ENUM** – это столбец, который может принимать значение из списка допустимых значений, явно перечисленных в спецификации столбца в момент создания таблицы.

Перечисление может иметь максимум 65535 элементов.

```
CREATE TABLE `t4`  
(  
    `en1` enum('orange','apple','grape') NOT NULL DEFAULT 'apple'  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
INSERT INTO `t4` (`en1`) VALUES ('grape');  
SELECT * from `t4`;
```

|       |
|-------|
| en1   |
| grape |

**SET** – строковый тип, который может принимать ноль или более значений (до 64), каждое из которых должно быть выбрано из списка допустимых значений, определённых при создании таблицы.

Концевые пробелы удаляются из значений множества SET в момент создания таблицы.

MySQL сохраняет значения SET в численном виде, где младший бит сохраненной величины соответствует первому элементу множества.

Если вы делаете выборку столбца SET в числовом контексте, полученное значение содержит соответствующим образом установленные биты, создающие значение столбца.

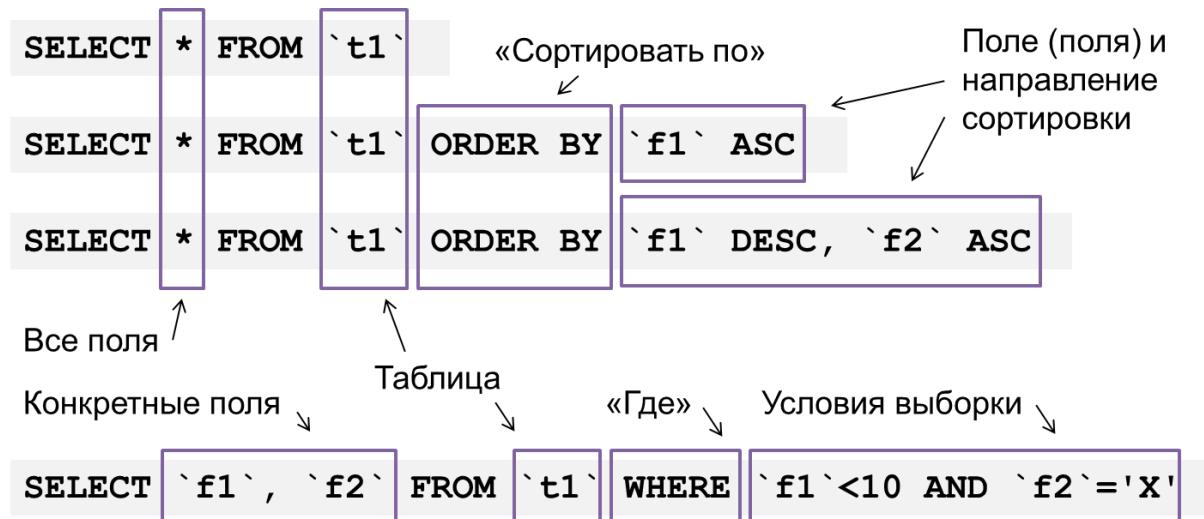
```
CREATE TABLE `t5` (
`se1` SET('read', 'write', 'delete') NOT NULL
) ENGINE = InnoDB;
INSERT INTO `t5` (`se1`) VALUES ('read,write');
SELECT * from `t5`;
```

|            |
|------------|
| se1        |
| read,write |

## 5.4. Элементарное управление данными в MySQL

### 5.4.1. Выборка данных

Упрощённая структура оператора SELECT такова:



### Элементарные примеры

Допустим, у нас есть такая таблица:

```

CREATE TABLE `payment` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) NOT NULL,
  `money` int(11) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8
    
```

Выбрать все записи и все поля (показать всю таблицу):

| Исходная таблица   | Запрос      | Результат    |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
|--|-------------|--------------|--------------|---|------|----|---|------|----|---|------|----|---|------|---|---|---------|-----|---|---------|-----|--------------------------------------|--|-----------|-------------|--------------|---|------|----|---|------|----|---|------|----|---|------|---|---|---------|-----|---|---------|-----|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr><td>1</td><td>John</td><td>10</td></tr> <tr><td>2</td><td>Jack</td><td>20</td></tr> <tr><td>3</td><td>Alex</td><td>30</td></tr> <tr><td>4</td><td>John</td><td>5</td></tr> <tr><td>5</td><td>Someone</td><td>999</td></tr> <tr><td>6</td><td>Someone</td><td>999</td></tr> </tbody> </table> | <b>id</b>   | <b>name</b>  | <b>money</b> | 1 | John | 10 | 2 | Jack | 20 | 3 | Alex | 30 | 4 | John | 5 | 5 | Someone | 999 | 6 | Someone | 999 | <code>SELECT * from `payment`</code> | <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr><td>1</td><td>John</td><td>10</td></tr> <tr><td>2</td><td>Jack</td><td>20</td></tr> <tr><td>3</td><td>Alex</td><td>30</td></tr> <tr><td>4</td><td>John</td><td>5</td></tr> <tr><td>5</td><td>Someone</td><td>999</td></tr> <tr><td>6</td><td>Someone</td><td>999</td></tr> </tbody> </table> | <b>id</b> | <b>name</b> | <b>money</b> | 1 | John | 10 | 2 | Jack | 20 | 3 | Alex | 30 | 4 | John | 5 | 5 | Someone | 999 | 6 | Someone | 999 |
| <b>id</b>  | <b>name</b> | <b>money</b> |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| 1  | John        | 10           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| 2  | Jack        | 20           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| 3  | Alex        | 30           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| 4  | John        | 5            |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| 5  | Someone     | 999          |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| 6  | Someone     | 999          |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| <b>id</b>  | <b>name</b> | <b>money</b> |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| 1  | John        | 10           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| 2  | Jack        | 20           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| 3  | Alex        | 30           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| 4  | John        | 5            |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| 5  | Someone     | 999          |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| 6  | Someone     | 999          |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |

Показать все имена без повторений:

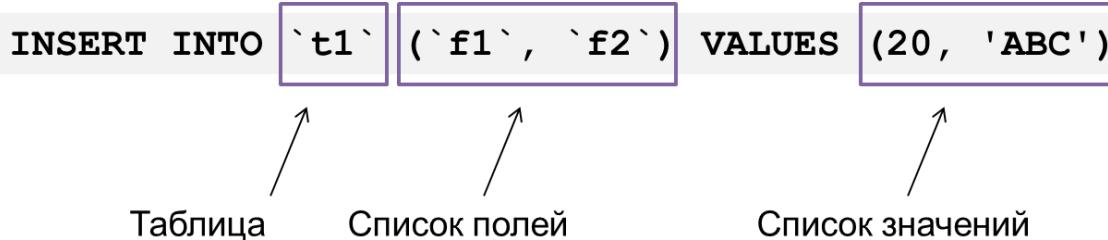
| Исходная таблица   | Запрос      | Результат    |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |   |             |      |      |      |         |
|--|-------------|--------------|--------------|---|------|----|---|------|----|---|------|----|---|------|---|---|---------|-----|---|---------|-----|--|---|-------------|------|------|------|---------|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>John</td><td>10</td></tr> <tr> <td>2</td><td>Jack</td><td>20</td></tr> <tr> <td>3</td><td>Alex</td><td>30</td></tr> <tr> <td>4</td><td>John</td><td>5</td></tr> <tr> <td>5</td><td>Someone</td><td>999</td></tr> <tr> <td>6</td><td>Someone</td><td>999</td></tr> </tbody> </table> | <b>id</b>   | <b>name</b>  | <b>money</b> | 1 | John | 10 | 2 | Jack | 20 | 3 | Alex | 30 | 4 | John | 5 | 5 | Someone | 999 | 6 | Someone | 999 | <code>SELECT DISTINCT `name` from `payment`</code> | <table border="1"> <thead> <tr> <th><b>name</b></th></tr> </thead> <tbody> <tr> <td>John</td></tr> <tr> <td>Jack</td></tr> <tr> <td>Alex</td></tr> <tr> <td>Someone</td></tr> </tbody> </table> | <b>name</b> | John | Jack | Alex | Someone |
| <b>id</b>  | <b>name</b> | <b>money</b> |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |   |             |      |      |      |         |
| 1  | John        | 10           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |   |             |      |      |      |         |
| 2  | Jack        | 20           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |   |             |      |      |      |         |
| 3  | Alex        | 30           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |   |             |      |      |      |         |
| 4  | John        | 5            |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |   |             |      |      |      |         |
| 5  | Someone     | 999          |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |   |             |      |      |      |         |
| 6  | Someone     | 999          |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |   |             |      |      |      |         |
| <b>name</b>  |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |   |             |      |      |      |         |
| John   |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |   |             |      |      |      |         |
| Jack   |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |   |             |      |      |      |         |
| Alex   |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |   |             |      |      |      |         |
| Someone  |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |   |             |      |      |      |         |

Показать сумму всех платежей:

| Исходная таблица   | Запрос      | Результат    |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                     |      |
|--|-------------|--------------|--------------|---|------|----|---|------|----|---|------|----|---|------|---|---|---------|-----|---|---------|-----|---|--|---------------------|------|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>John</td><td>10</td></tr> <tr> <td>2</td><td>Jack</td><td>20</td></tr> <tr> <td>3</td><td>Alex</td><td>30</td></tr> <tr> <td>4</td><td>John</td><td>5</td></tr> <tr> <td>5</td><td>Someone</td><td>999</td></tr> <tr> <td>6</td><td>Someone</td><td>999</td></tr> </tbody> </table> | <b>id</b>   | <b>name</b>  | <b>money</b> | 1 | John | 10 | 2 | Jack | 20 | 3 | Alex | 30 | 4 | John | 5 | 5 | Someone | 999 | 6 | Someone | 999 | <code>SELECT SUM(`money`) from `payment`</code> | <table border="1"> <thead> <tr> <th><b>SUM(`money`)</b></th></tr> </thead> <tbody> <tr> <td>2063</td></tr> </tbody> </table> | <b>SUM(`money`)</b> | 2063 |
| <b>id</b>  | <b>name</b> | <b>money</b> |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                     |      |
| 1  | John        | 10           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                     |      |
| 2  | Jack        | 20           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                     |      |
| 3  | Alex        | 30           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                     |      |
| 4  | John        | 5            |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                     |      |
| 5  | Someone     | 999          |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                     |      |
| 6  | Someone     | 999          |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                     |      |
| <b>SUM(`money`)</b>  |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                     |      |
| 2063   |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                     |      |

## 5.4.2. Вставка данных

Упрощённая структура оператора INSERT такова:



### Элементарный пример

Добавить новый платёж:

```
INSERT INTO `payment` (`id`, `name`, `money`)
VALUES (NULL, 'userN', 99)
```

## 5.4.3. Удаление данных

Упрощённая структура оператора DELETE такова:



### Элементарный пример

Удалить все платежи, выполненные пользователем 'Someone':

```
DELETE FROM `payment` WHERE `name` = 'Someone'
```

## 5.5. Основы использования MySQL Workbench

**MySQL Workbench** – бесплатная среда для управления MySQL.

Где взять: <http://www.mysql.com/downloads/workbench/>

Преимущества:

- легко установить;
- просто изучить.

Недостатки:

- работает на стороне клиента;
- часть функционала избыточна для «повседневного использования».

Здесь мы рассмотрим исключительно **основы** работы с MySQL Workbench.

Более подробное руководство можно найти здесь:

<http://dev.mysql.com/doc/workbench/en/index.html>

Некоторое количество видео-пособий можно найти здесь:

<http://forums.mysql.com/read.php?152,283061,283061>

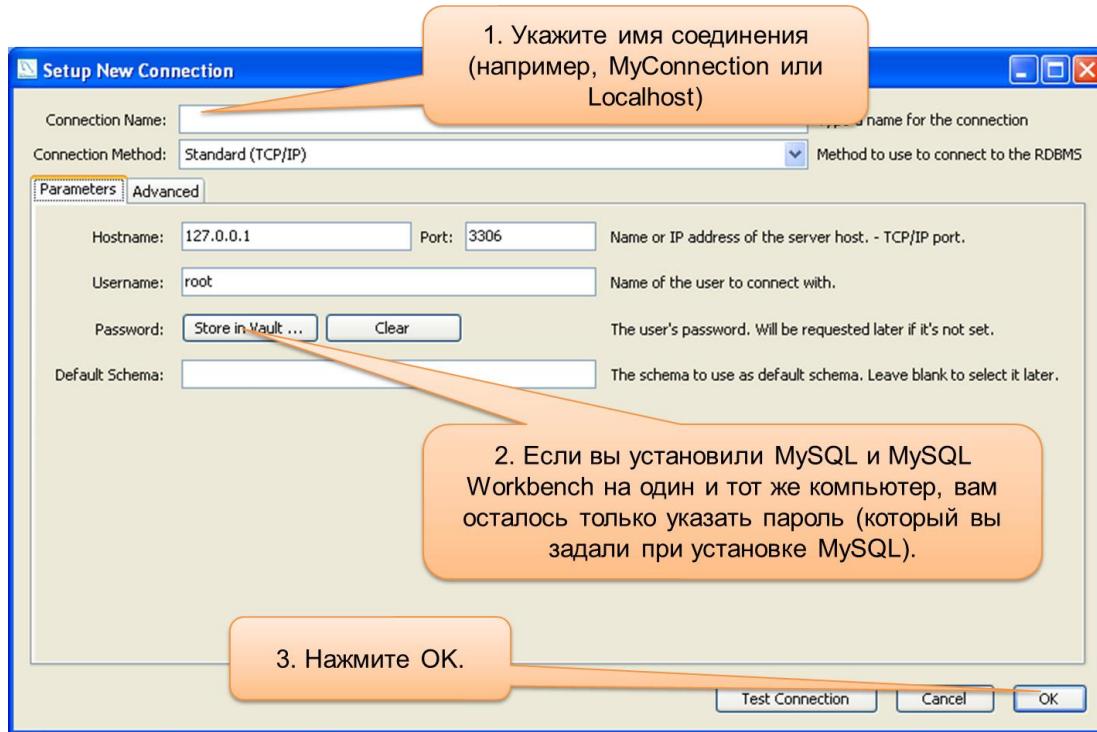
MySQL Workbench устанавливается как обычное приложение, потому процесс установки мы рассматривать не будем в силу его простоты.

### Создание соединения

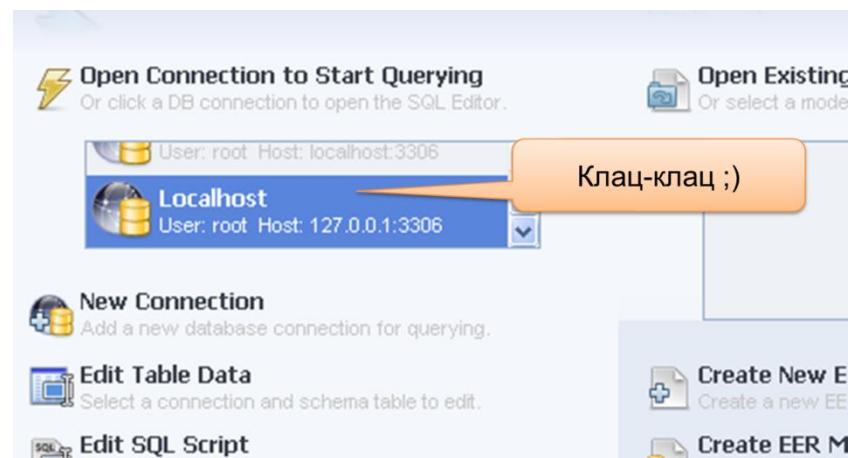
После первого запуска MySQL Workbench рекомендуется создать соединение с СУБД, которое в будущем мы и будем использовать:



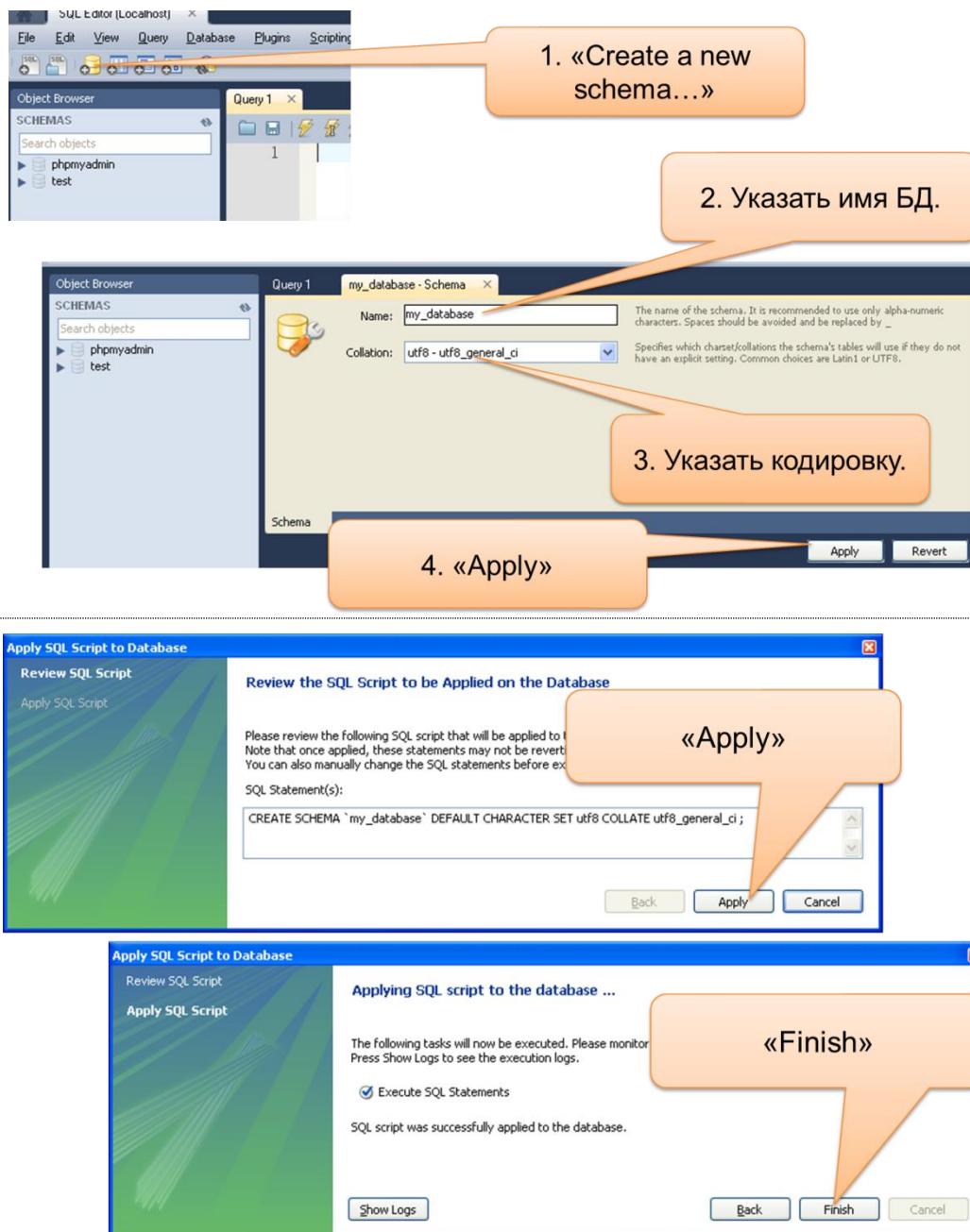
Теперь надо выполнить следующие шаги.



Двойной клик по только что созданному соединению запускает сессию с MySQL.



Приступаем к созданию базы данных.



Остается только...



Теперь можно создать таблицу.

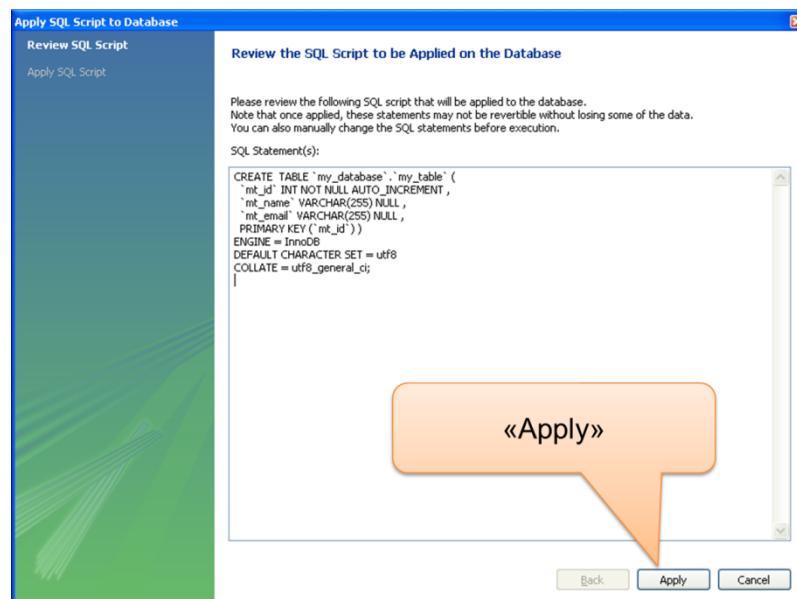
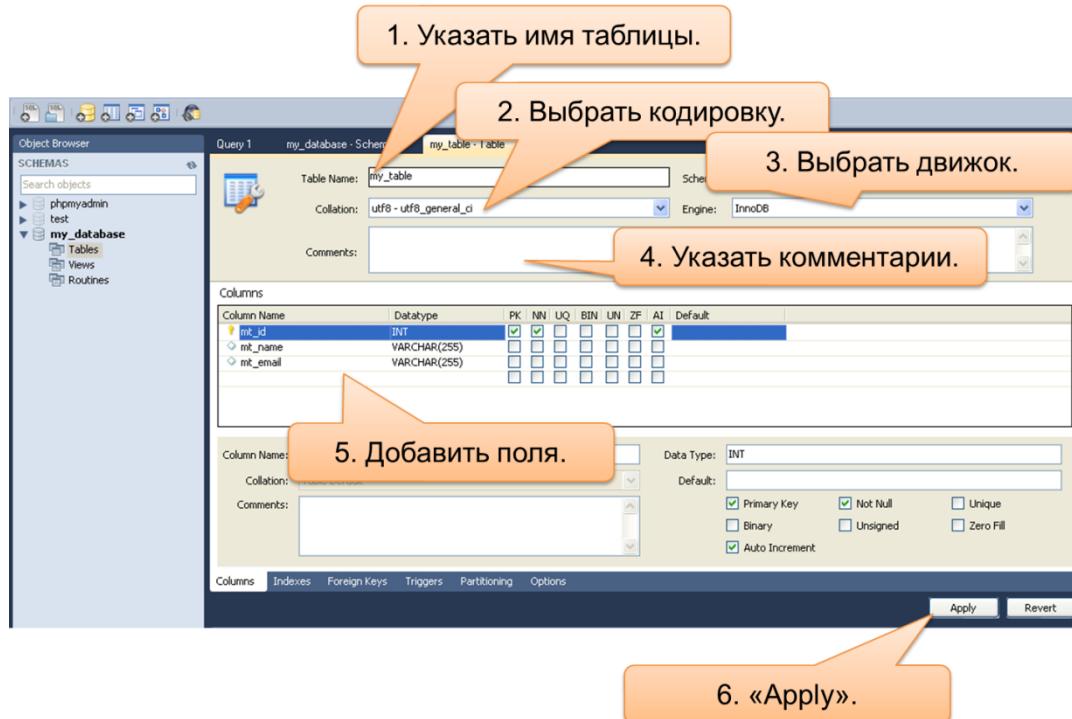
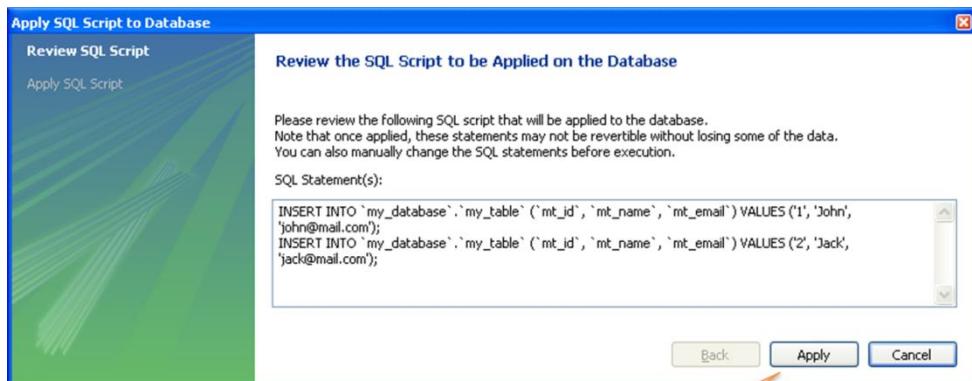
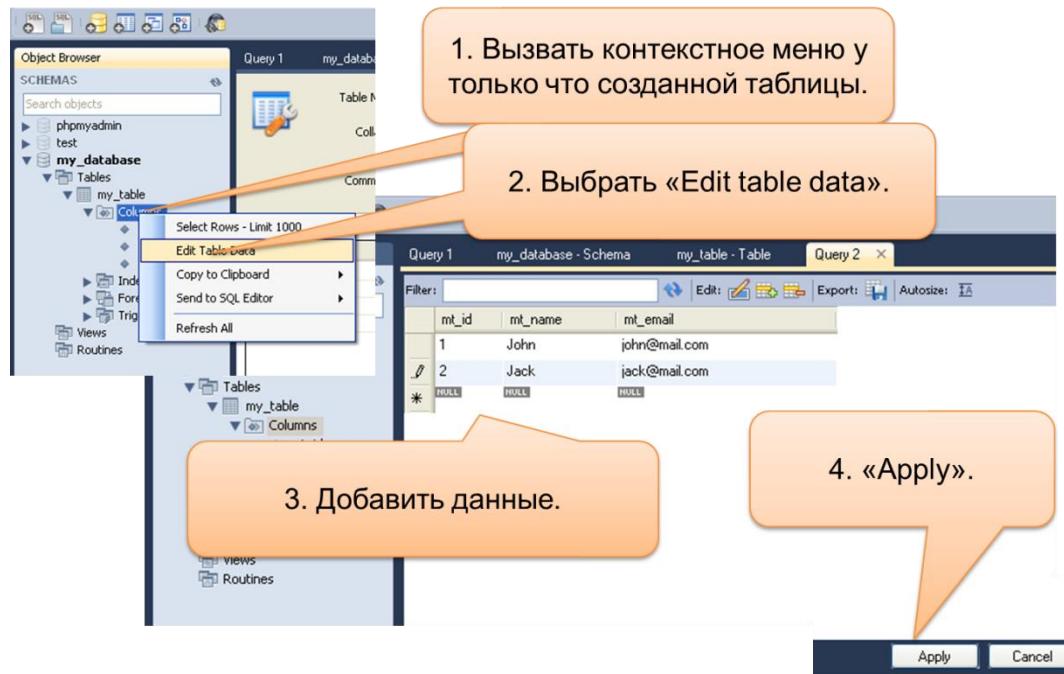
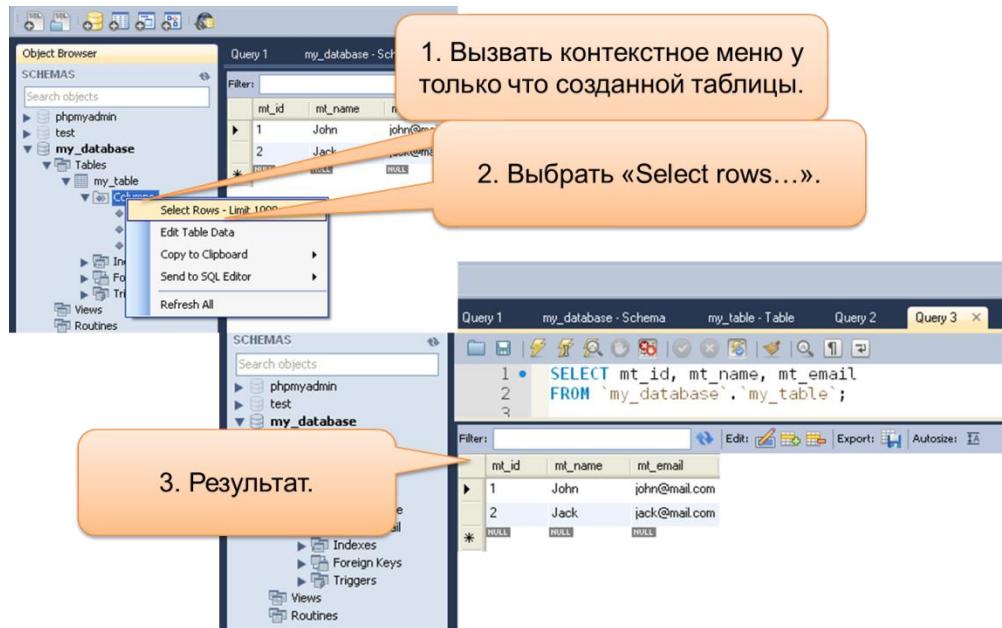


Таблица готова. Можно наполнять её данными.

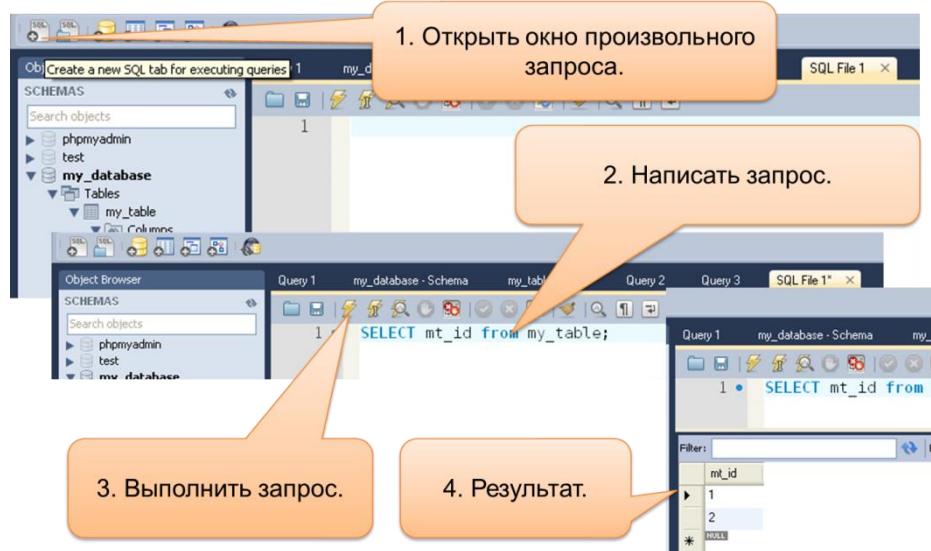
## Добавляем данные.



Можно просмотреть данные таблицы.



Для выполнения произвольного запроса нужно сделать следующее.



### Краткое заключение

У MySQL Workbench очень богатый функционал (в частности, мы не затронули вопрос разработки модели БД и администрирования СУБД).

Мы не можем (и не будем!) дублировать здесь полное руководство по MySQL Workbench. Его можно изучить самостоятельно здесь:

<http://dev.mysql.com/doc/workbench/en/index.html>

А мы переходим к рассмотрению основ работы с phpMyAdmin.

## 5.6. Основы использования phpMyAdmin

**phpMyAdmin** – бесплатная (и, пожалуй, самая распространённая) среда для управления MySQL.

Где взять: <http://www.phpmyadmin.net>

## Преимущества:

- оптимальный для повседневных задач набор функционала;
  - работает на стороне сервера (через браузер).

## Недостатки:

- тонкая настройка требует знаний и умений;
  - не предоставляет столь же полного набора возможностей, как MySQL Workbench.

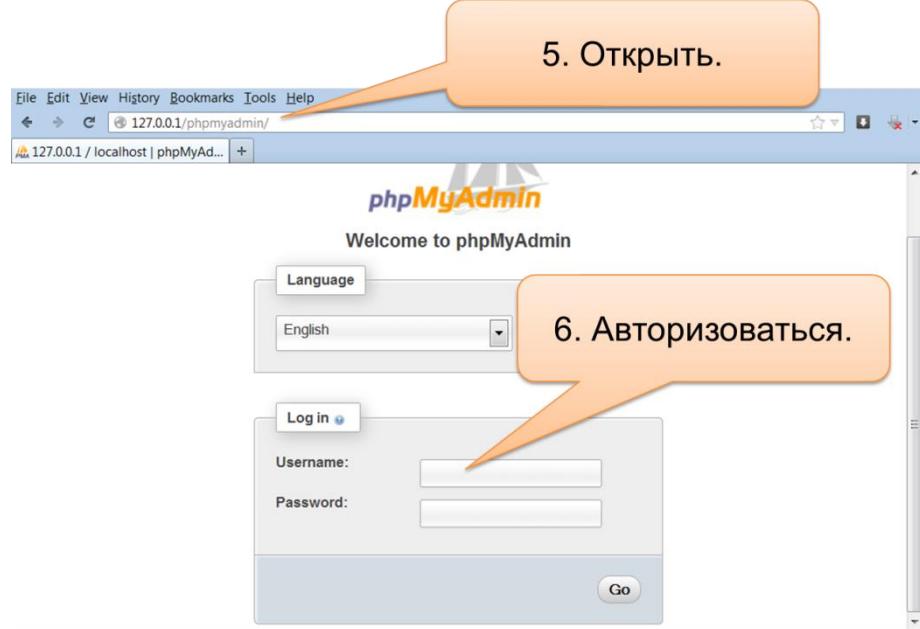
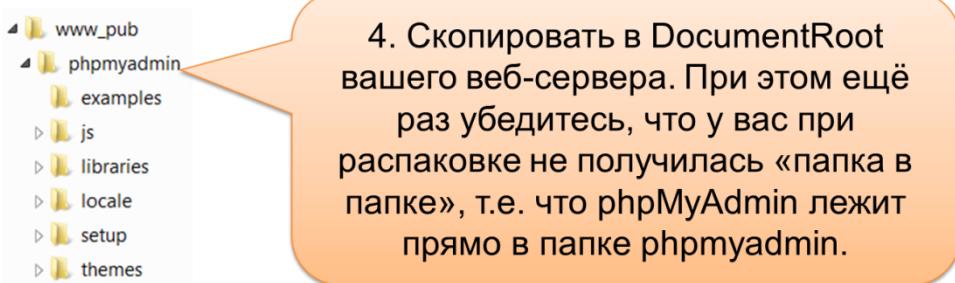
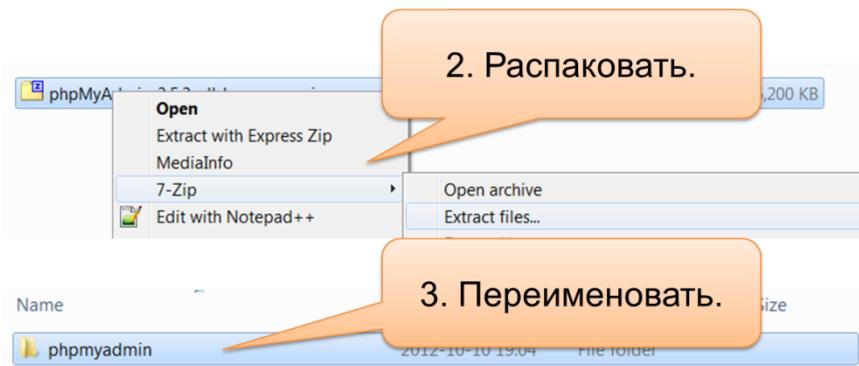
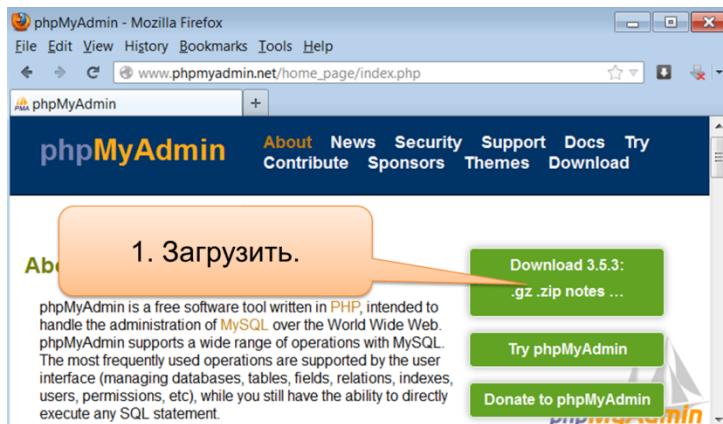
Подробнее о документации по phpMyAdmin можно взять здесь:

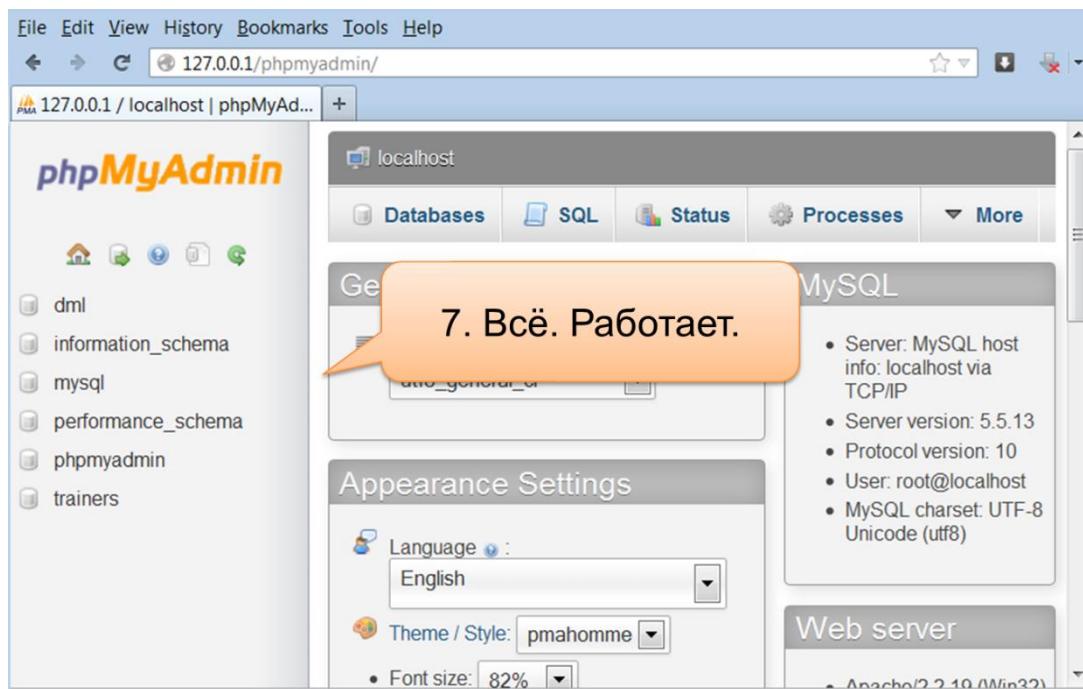
[http://www.phpmyadmin.net/home\\_page/docs.php](http://www.phpmyadmin.net/home_page/docs.php)

Для работы phpMyAdmin требуется веб-сервер (например, Apache) и PHP. Поскольку «объять необъятное» невозможно, будем считать, что Apache+MySQL+PHP уже установлены и настроены.

Мы же рассмотрим установку и настройку самого phpMyAdmin, а также выполнение с его помощью некоторых базовых операций.

## Установка





## Настройка



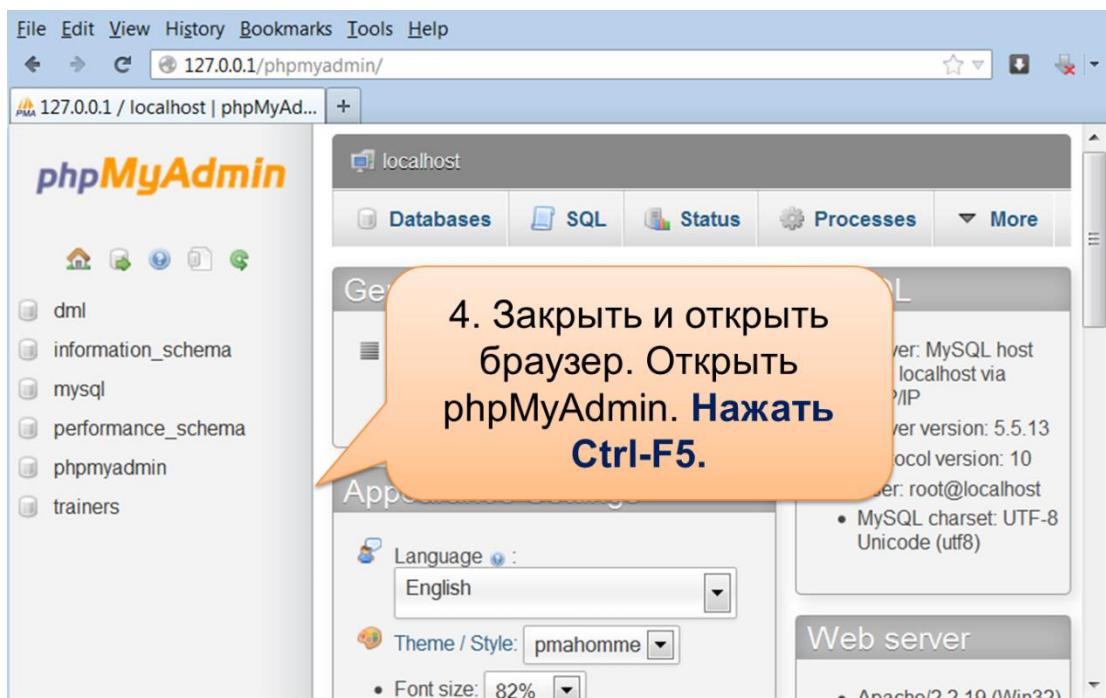
1. В папке phpmyadmin файл config.sample.inc.php скопировать под именем config.inc.php.

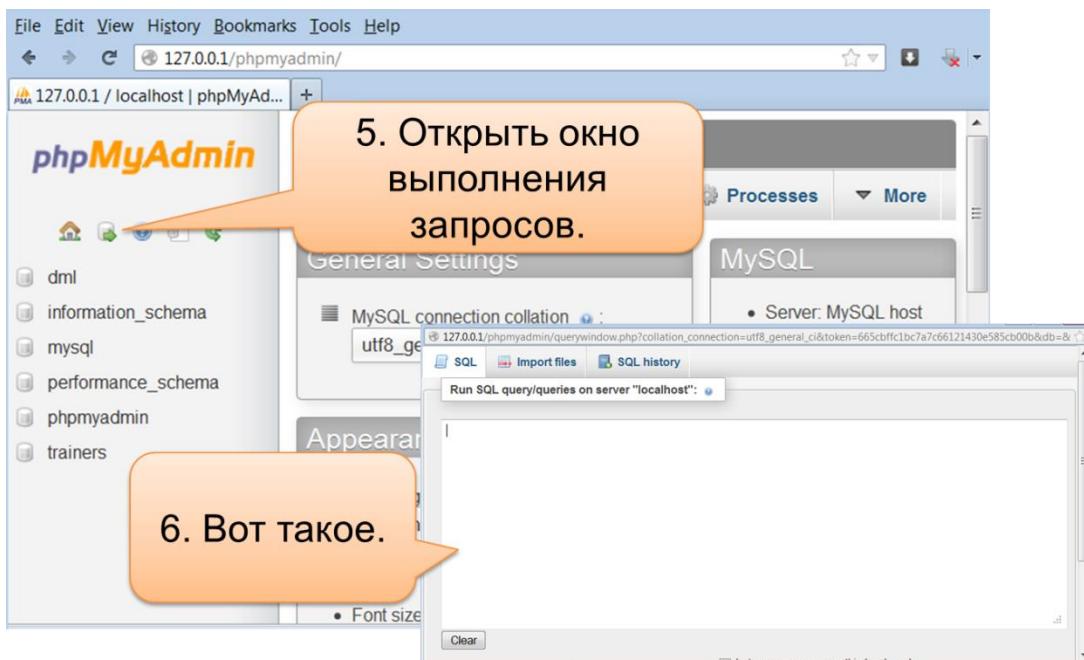
2. В секции «Authentication type» файла config.inc.php установить вот такие данные:

```
/* Authentication type */
$cfg['Servers'][$i]['auth_type'] = 'config';
$cfg['Servers'][$i]['user'] = 'ваш_логин_к_MySQL';
$cfg['Servers'][$i]['password'] = 'ваш_пароль_к_MySQL';
```

3. В этом же файле (config.inc.php) полностью раскомментировать секцию «Storage database and tables»:

```
/* Storage database and tables */
$cfg['Servers'][$i]['pmadb'] = 'phpmyadmin';
$cfg['Servers'][$i]['bookmarktable'] = 'pma_bookmark';
$cfg['Servers'][$i]['relation'] = 'pma_relation';
$cfg['Servers'][$i]['table_info'] = 'pma_table_info';
$cfg['Servers'][$i]['table_coords'] = 'pma_table_coords';
$cfg['Servers'][$i]['pdf_pages'] = 'pma_pdf_pages';
$cfg['Servers'][$i]['column_info'] = 'pma_column_info';
$cfg['Servers'][$i]['history'] = 'pma_history';
$cfg['Servers'][$i]['tracking'] = 'pma_tracking';
$cfg['Servers'][$i]['designer_coords'] = 'pma_designer_coords';
$cfg['Servers'][$i]['userconfig'] = 'pma_userconfig';
```





Теперь надо выполнить вот эти два запроса.

Первый (о нём забывают во многих инструкциях):  
`create user 'pma'@'localhost' identified by 'pmapass';`

Второй (теоретически, такой запрос лежит в папке scripts в дистрибутиве phpMyAdmin, но практика показала, что «бывают разные случаи», а потому – вот текст точно работающего скрипта):

```
DROP DATABASE IF EXISTS `phpmyadmin`;
CREATE DATABASE `phpmyadmin`;
USE phpmyadmin;

-----
-- Privileges
--
GRANT SELECT, INSERT, DELETE, UPDATE ON `phpmyadmin`.* TO
    'pma'@localhost;
--

-- Table structure for table `pma_bookmark`
--

CREATE TABLE `pma_bookmark` (
    `id` int(11) NOT NULL auto_increment,
    `dbase` varchar(255) NOT NULL default '',
    `user` varchar(255) NOT NULL default '',
    `label` varchar(255) NOT NULL default '',
    `query` text NOT NULL,
    PRIMARY KEY  (`id`)
) ENGINE=MyISAM COMMENT='Bookmarks';
--
```

```
--  
-- Table structure for table `pma_column_info`  
  
CREATE TABLE `pma_column_info` (  
  `id` int(5) unsigned NOT NULL auto_increment,  
  `db_name` varchar(64) NOT NULL default '',  
  `table_name` varchar(64) NOT NULL default '',  
  `column_name` varchar(64) NOT NULL default '',  
  `comment` varchar(255) NOT NULL default '',  
  `mimetype` varchar(255) NOT NULL default '',  
  `transformation` varchar(255) NOT NULL default '',  
  `transformation_options` varchar(255) NOT NULL default '',  
  PRIMARY KEY (`id`),  
  UNIQUE KEY `db_name` (`db_name`, `table_name`, `column_name`)  
) ENGINE=MyISAM COMMENT='Column information for phpMyAdmin';  
  
-----  
  
--  
-- Table structure for table `pma_history`  
  
CREATE TABLE `pma_history` (  
  `id` bigint(20) unsigned NOT NULL auto_increment,  
  `username` varchar(64) NOT NULL default '',  
  `db` varchar(64) NOT NULL default '',  
  `table` varchar(64) NOT NULL default '',  
  `timevalue` timestamp,  
  `sqlquery` text NOT NULL,  
  PRIMARY KEY (`id`),  
  KEY `username` (`username`, `db`, `table`, `timevalue`)  
) ENGINE=MyISAM COMMENT='SQL history for phpMyAdmin';  
  
-----  
  
--  
-- Table structure for table `pma_pdf_pages`  
  
CREATE TABLE `pma_pdf_pages` (  
  `db_name` varchar(64) NOT NULL default '',  
  `page_nr` int(10) unsigned NOT NULL auto_increment,  
  `page_descr` varchar(50) NOT NULL default '',  
  PRIMARY KEY (`page_nr`),  
  KEY `db_name` (`db_name`)  
) ENGINE=MyISAM COMMENT='PDF relation pages for phpMyAdmin';  
  
-----  
  
--  
-- Table structure for table `pma_relation`  
  
CREATE TABLE `pma_relation` (  
  `master_db` varchar(64) NOT NULL default '',  
  `master_table` varchar(64) NOT NULL default '',  
  `master_field` varchar(64) NOT NULL default '',  
  `foreign_db` varchar(64) NOT NULL default '',  
  `foreign_table` varchar(64) NOT NULL default '',  
  `foreign_field` varchar(64) NOT NULL default '',  
  PRIMARY KEY (`master_db`, `master_table`, `master_field`),  
  KEY `foreign_field` (`foreign_db`, `foreign_table`)
```

```
) ENGINE=MyISAM COMMENT='Relation table';

-----
-- Table structure for table `pma_table_coords`
--

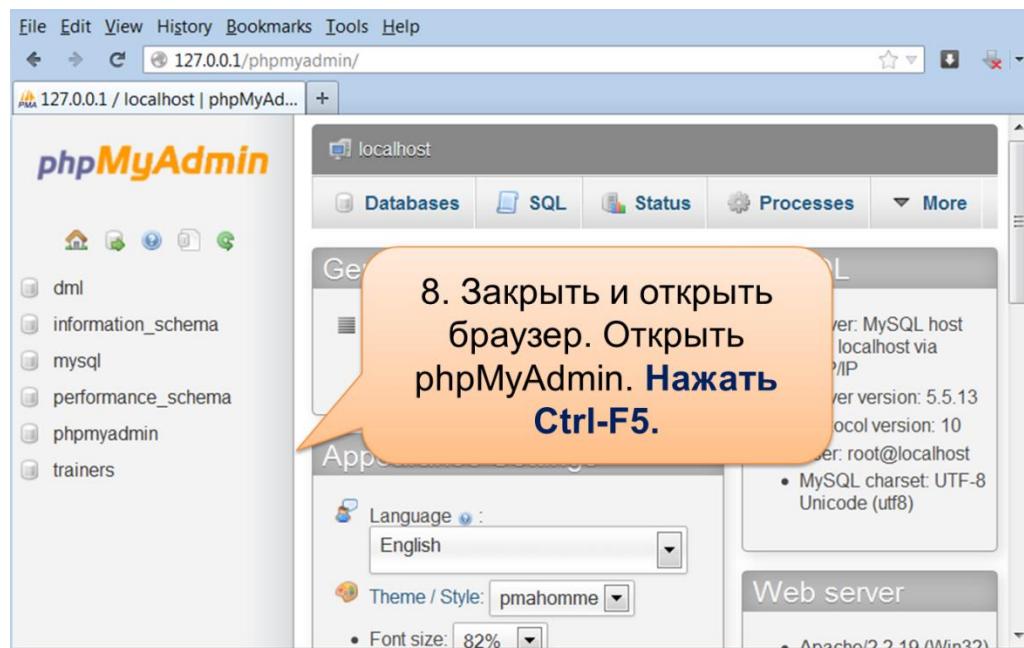
CREATE TABLE `pma_table_coords` (
  `db_name` varchar(64) NOT NULL default '',
  `table_name` varchar(64) NOT NULL default '',
  `pdf_page_number` int(11) NOT NULL default '0',
  `x` float unsigned NOT NULL default '0',
  `y` float unsigned NOT NULL default '0',
  PRIMARY KEY  (`db_name`, `table_name`, `pdf_page_number`)
) ENGINE=MyISAM COMMENT='Table coordinates for phpMyAdmin PDF output';

-----
-- Table structure for table `pma_table_info`
--

CREATE TABLE `pma_table_info` (
  `db_name` varchar(64) NOT NULL default '',
  `table_name` varchar(64) NOT NULL default '',
  `display_field` varchar(64) NOT NULL default '',
  PRIMARY KEY  (`db_name`, `table_name`)
) ENGINE=MyISAM COMMENT='Table information for phpMyAdmin';

-----
-- Table structure for table `pma_designer_coords`
--

CREATE TABLE `pma_designer_coords` (
  `db_name` varchar(64) NOT NULL default '',
  `table_name` varchar(64) NOT NULL default '',
  `x` INT(11) default NULL,
  `y` INT(11) default NULL,
  `v` TINYINT(4) default NULL,
  `h` TINYINT(4) default NULL,
  PRIMARY KEY (`db_name`, `table_name`)
) ENGINE=MyISAM COMMENT='Table coordinates for Designer'
```



9. Выбрать любую непустую БД.

10. Выбрать «**Designer**».

11. Если вы всё сделали правильно, вы увидите примерно такое.

## Использование

Подробнейшие видео-пособия по phpMyAdmin можно найти здесь:  
<http://support247webs.com/phpmyadmin-tutorials.htm>

Также мы будем пользоваться, в основном, phpMyAdmin и консолью MySQL в дальнейших темах, т.е. сможем увидеть всё в действии.

Потому на этом с данной темой – ВСЁ ☺.

## Тема 6: Язык управления данными в MySQL

### 6.1. Оператор SELECT, общая структура

#### 6.1.1. Обзор структуры и краткие пояснения

Поскольку общая структура оператора SELECT достаточно обширна, в начале каждого подраздела будет приведена небольшая картинка, на которой будет отмечено, какую часть структуры мы сейчас будем рассматривать.

Итак, вся структура оператора SELECT выглядит следующим образом:

```
SELECT
  [ALL | DISTINCT | DISTINCTROW ]
  [HIGH_PRIORITY]
  [STRAIGHT_JOIN]
  [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
  [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
  select_expr [, select_expr ...]
  [FROM table_references
  [WHERE where_condition]
  [GROUP BY {col_name | expr | position}
    [ASC | DESC], ... [WITH ROLLUP]]
  [HAVING where_condition]
  [ORDER BY {col_name | expr | position}
    [ASC | DESC], ...]
  [LIMIT {[offset,] row_count | row_count OFFSET offset}]
  [PROCEDURE procedure_name(argument_list)]
  [INTO OUTFILE 'file_name'
    [CHARACTER SET charset_name]
    export_options
    | INTO DUMPFILE 'file_name'
    | INTO var_name [, var_name]]
  [FOR UPDATE | LOCK IN SHARE MODE]]
```

#### 6.1.2. SELECT: [ALL | DISTINCT | DISTINCTROW ]

**SELECT**

**[ALL | DISTINCT | DISTINCTROW ]**

```
[HIGH_PRIORITY]
[STRAIGHT_JOIN]
[SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
[SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
select_expr [, select_expr ...]
[FROM table_references
[WHERE where_condition]
[GROUP BY {col_name | expr | position}
  [ASC | DESC], ... [WITH ROLLUP]]
[HAVING where_condition]
[ORDER BY {col_name | expr | position}
  [ASC | DESC], ...]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
[PROCEDURE procedure_name(argument_list)]
[INTO OUTFILE 'file_name'
  [CHARACTER SET charset_name]
  export_options
  | INTO DUMPFILE 'file_name'
  | INTO var_name [, var_name]]
[FOR UPDATE | LOCK IN SHARE MODE]]
```

**ALL** – вернуть все (в т.ч. повторяющиеся) ряды (значение по умолчанию).

**DISTINCT** (синоним DISTINCTROW) – вернуть только уникальные ряды (популярные игнорировать).

Одновременно указывать ALL и DISTINCT нельзя.

Для такой таблицы следующие запросы вернут одинаковые результаты (все ряды), т.к. дублированных рядов здесь нет.

| Исходная таблица  | Запрос                                    | Результат   |      |         |      |       |                                      |   |      |         |      |       |      |       |
|---|---|---|------|---------|------|-------|--------------------------------------|---|------|---------|------|-------|------|-------|
|   | <code>SELECT * FROM `tbl`</code>          | <table border="1"> <thead> <tr> <th>name</th><th>surname</th></tr> </thead> <tbody> <tr> <td>John</td><td>Smith</td></tr> <tr> <td>John</td><td>Black</td></tr> </tbody> </table> | name | surname | John | Smith | John                                 | Black   |      |         |      |       |      |       |
| name  | surname                                   |   |      |         |      |       |                                      |   |      |         |      |       |      |       |
| John  | Smith                                     |   |      |         |      |       |                                      |   |      |         |      |       |      |       |
| John  | Black                                     |   |      |         |      |       |                                      |   |      |         |      |       |      |       |
| <table border="1"> <thead> <tr> <th>name</th><th>surname</th></tr> </thead> <tbody> <tr> <td>John</td><td>Smith</td></tr> <tr> <td>John</td><td>Black</td></tr> </tbody> </table> | name                                      | surname   | John | Smith   | John | Black | <code>SELECT ALL * FROM `tbl`</code> | <table border="1"> <thead> <tr> <th>name</th><th>surname</th></tr> </thead> <tbody> <tr> <td>John</td><td>Smith</td></tr> <tr> <td>John</td><td>Black</td></tr> </tbody> </table> | name | surname | John | Smith | John | Black |
| name  | surname                                   |   |      |         |      |       |                                      |   |      |         |      |       |      |       |
| John  | Smith                                     |   |      |         |      |       |                                      |   |      |         |      |       |      |       |
| John  | Black                                     |   |      |         |      |       |                                      |   |      |         |      |       |      |       |
| name  | surname                                   |   |      |         |      |       |                                      |   |      |         |      |       |      |       |
| John  | Smith                                     |   |      |         |      |       |                                      |   |      |         |      |       |      |       |
| John  | Black                                     |   |      |         |      |       |                                      |   |      |         |      |       |      |       |
|   | <code>SELECT DISTINCT * FROM `tbl`</code> | <table border="1"> <thead> <tr> <th>name</th><th>surname</th></tr> </thead> <tbody> <tr> <td>John</td><td>Smith</td></tr> <tr> <td>John</td><td>Black</td></tr> </tbody> </table> | name | surname | John | Smith | John                                 | Black   |      |         |      |       |      |       |
| name  | surname                                   |   |      |         |      |       |                                      |   |      |         |      |       |      |       |
| John  | Smith                                     |   |      |         |      |       |                                      |   |      |         |      |       |      |       |
| John  | Black                                     |   |      |         |      |       |                                      |   |      |         |      |       |      |       |

Но при выборе только первой колонки есть различия:

| Исходная таблица  | Запрос                                    | Результат   |      |       |      |       |  |   |      |      |
|---|---|---|------|-------|------|-------|--|---|------|------|
|   | <code>SELECT ALL `name` FROM `tbl`</code> | <table border="1"> <thead> <tr> <th>name</th></tr> </thead> <tbody> <tr> <td>John</td></tr> <tr> <td>John</td></tr> </tbody> </table> | name | John  | John |       |  |   |      |      |
| name  |   |   |      |       |      |       |  |   |      |      |
| John  |   |   |      |       |      |       |  |   |      |      |
| John  |   |   |      |       |      |       |  |   |      |      |
| <table border="1"> <thead> <tr> <th>name</th><th>surname</th></tr> </thead> <tbody> <tr> <td>John</td><td>Smith</td></tr> <tr> <td>John</td><td>Black</td></tr> </tbody> </table> | name                                      | surname   | John | Smith | John | Black | <code>SELECT DISTINCT `name` FROM `tbl`</code> | <table border="1"> <thead> <tr> <th>name</th></tr> </thead> <tbody> <tr> <td>John</td></tr> </tbody> </table> | name | John |
| name  | surname                                   |   |      |       |      |       |  |   |      |      |
| John  | Smith                                     |   |      |       |      |       |  |   |      |      |
| John  | Black                                     |   |      |       |      |       |  |   |      |      |
| name  |   |   |      |       |      |       |  |   |      |      |
| John  |   |   |      |       |      |       |  |   |      |      |

Чуть более необычная ситуация: допустим, мы хотим извлечь имена без повторений, а также фамилии.

| Исходная таблица  | Запрос   | Результат   |      |         |      |       |                     |       |
|---|--|---|------|---------|------|-------|---------------------|-------|
|   | <code>SELECT DISTINCT(`name`), `surname` FROM `tbl`</code> | <table border="1"> <thead> <tr> <th>name</th><th>surname</th></tr> </thead> <tbody> <tr> <td>John</td><td>Smith</td></tr> <tr> <td>John</td><td>Black</td></tr> </tbody> </table> | name | surname | John | Smith | John                | Black |
| name  | surname  |   |      |         |      |       |                     |       |
| John  | Smith  |   |      |         |      |       |                     |       |
| John  | Black  |   |      |         |      |       |                     |       |
| <table border="1"> <thead> <tr> <th>name</th><th>surname</th></tr> </thead> <tbody> <tr> <td>John</td><td>Smith</td></tr> <tr> <td>John</td><td>Black</td></tr> </tbody> </table> | name   | surname   | John | Smith   | John | Black | <b>НЕ РАБОТАЕТ!</b> |       |
| name  | surname  |   |      |         |      |       |                     |       |
| John  | Smith  |   |      |         |      |       |                     |       |
| John  | Black  |   |      |         |      |       |                     |       |

**Q:** Когда ALL (значение по умолчанию) надо указывать явно?

**A:** В случае с использованием UNION.

Допустим, мы хотим узнать, кто из пользователей внес меньше всего и больше всего денег. Пусть у нас есть такая таблица.

| Исходная таблица |                |              | Запрос   | Результат           |
|------------------|----------------|--------------|--|---------------------|
| <b>name</b>      | <b>surname</b> | <b>money</b> |  |                     |
| John             | Smith          | 50           | (SELECT `name` FROM `t_all_distinct` ORDER BY `money` ASC LIMIT 1)<br>UNION<br>(SELECT `name` FROM `t_all_distinct` ORDER BY `money` DESC LIMIT 1)     | <b>name</b><br>John |
| John             | Black          | 50           | (SELECT `name` FROM `t_all_distinct` ORDER BY `money` ASC LIMIT 1)<br>UNION ALL<br>(SELECT `name` FROM `t_all_distinct` ORDER BY `money` DESC LIMIT 1) | <b>name</b><br>John |
| John             | Black          | 100          |  | John                |

Можно ли как-то улучшить эти запросы (сделать их проще, быстрее, логичнее)? Как насчёт идеи с MAX() и MIN() вместо ORDER BY?

| Исходная таблица |                |              | Запрос  | Результат                    |
|------------------|----------------|--------------|---|------------------------------|
| <b>name</b>      | <b>surname</b> | <b>money</b> |   |                              |
| John             | Smith          | 50           | (SELECT `name`, `money` FROM `t_all_distinct` HAVING `money` = MAX(`money`))<br>UNION ALL<br>(SELECT `name`, `money` FROM `t_all_distinct` HAVING `money` = MIN(`money`)) | <b>name money</b><br>John 50 |
| John             | Black          | 50           |   | Странно, да?!                |
| John             | Black          | 100          |   |                              |

Это – особенность MySQL. Другие СУБД могут вообще не выполнить такой запрос, а MySQL выполняет его с такой ошибкой.

Подробности – здесь: <http://stackoverflow.com/questions/8072231/mysql-how-to-select-minium-and-maximum-in-one-union-query>

И что делать?! Использовать подзапросы.

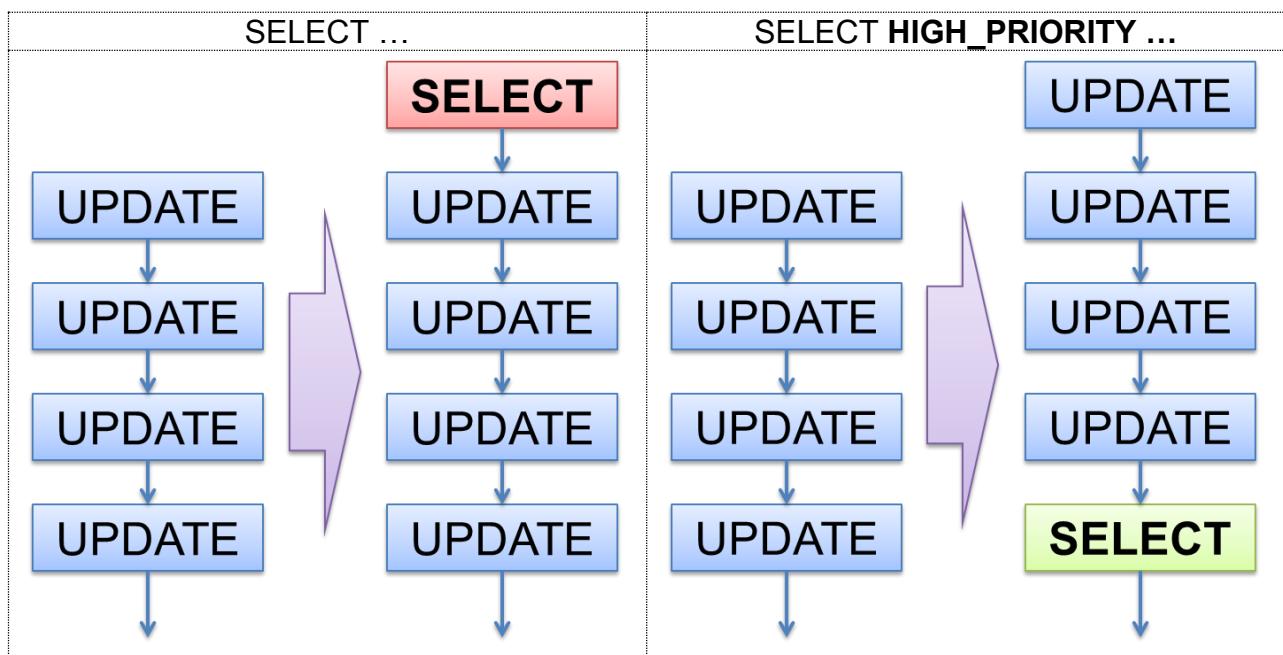
| Исходная таблица |                |              | Запрос  | Результат                     |
|------------------|----------------|--------------|---|-------------------------------|
| <b>name</b>      | <b>surname</b> | <b>money</b> |   |                               |
| John             | Smith          | 50           | (SELECT `name`, `money` FROM `t_all_distinct` WHERE `money` = (SELECT MAX(`money`) FROM `t_all_distinct`))<br>LIMIT 1)<br>UNION ALL<br>(SELECT `name`, `money` FROM `t_all_distinct` WHERE `money` = (SELECT MIN(`money`) FROM `t_all_distinct`))<br>LIMIT 1) | <b>name money</b><br>John 100 |
| John             | Black          | 50           |   | John 50                       |
| John             | Black          | 100          |   |                               |

### 6.1.3. SELECT: [HIGH\_PRIORITY]

```
SELECT
[ALL | DISTINCT | DISTINCTROW ]
[HIGH_PRIORITY]
[STRAIGHT_JOIN]
[SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
[SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
select_expr [, select_expr ...]
[FROM table_references
[WHERE where_condition]
[GROUP BY (col_name | expr | position)
[ASC | DESC], ... [WITH ROLLUP]]
[HAVING where_condition]
[ORDER BY (col_name | expr | position)
[ASC | DESC], ...]
[LIMIT {offset}, row_count | row_count OFFSET offset]
[PROCEDURE procedure_name(argument_list)]
[INTO OUTFILE 'file_name'
[CHARACTER SET charset_name]
export_options
| INTO DUMPFILE 'file_name'
| INTO var_name [, var_name]
| FOR UPDATE | LOCK IN SHARE MODE]]
```

Использование ключевого слова **HIGH\_PRIORITY** позволяет при работе с таблицами на движках MyISAM, MEMORY и MERGE выставить SELECT-запросу более высокий приоритет, чем модифицирующим запросам (INSERT, UPDATE, DELETE, REPLACE).

Как это работает:



В руководстве сказано...

Использовать HIGH\_PRIORITY только с очень быстрыми запросами, которые обязательно надо выполнить прямо сейчас.

Использование HIGH\_PRIORITY недопустимо с запросами в контексте UNION.

Использование ключевого слова **HIGH\_PRIORITY** не имеет смысла с движком InnoDB, в котором используются блокировки уровня записи и «сообщения о намерении» выставить блокировку»:

- Блокировка S (shared) – чтение записей.
- Блокировка X (exclusive) – модификация записей.
- Блокировка IS (intension shared) – транзакция собирается получить блокировку S.
- Блокировка IX (intension exclusive) – транзакция собирается получить блокировку X.

## Сводная таблица совместимости блокировок InnoDB:

|    | X        | IX         | S          | IS         |
|----|----------|------------|------------|------------|
| X  | Conflict | Conflict   | Conflict   | Conflict   |
| IX | Conflict | Compatible | Conflict   | Compatible |
| S  | Conflict | Conflict   | Compatible | Compatible |
| IS | Conflict | Compatible | Compatible | Compatible |

Подробно обо всём этом можно прочитать здесь:

[http://docs.oracle.com/cd/E17952\\_01/refman-5.5-en/innodb-lock-modes.html](http://docs.oracle.com/cd/E17952_01/refman-5.5-en/innodb-lock-modes.html)

### 6.1.4. SELECT: [STRAIGHT\_JOIN]

```
SELECT
[ALL | DISTINCT | DISTINCTROW ]
[HIGH_PRIORITY]
[STRAIGHT_JOIN]
[SQL_SMALL_RESULT | SQL_BIG_RESULT | SQL_BUFFER_RESULT]
[SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
select_expr [, select_expr ...]
[FROM table_references
[WHERE where_condition]
[GROUP BY (col_name | expr | position)
[ASC | DESC] ... [WITH ROLLUP]]
[HAVING where_condition]
[ORDER BY (col_name | expr | position)
[ASC | DESC] ...]
[LIMIT {offset}, row_count | row_count OFFSET offset]
[PROCEDURE procedure_name(argument_list)]
[INTO OUTFILE 'file_name'
[CHARACTER SET charset_name]
export_options
| INTO DUMPFILE 'file_name'
| INTO var_name [, var_name]
| FOR UPDATE | LOCK IN SHARE MODE]]
```

Использование ключевого слова **STRAIGHT\_JOIN** позволяет указать оптимизатору запросов, что в JOIN-запросе таблицы надо обрабатывать в том порядке, в котором они перечислены в списке таблиц в SELECT.

В таком запросе оптимизатор теоретически может сначала прочитать вторую таблицу:

```
SELECT * FROM `firstT` JOIN `secondT`
ON `firstT`.`id` = `secondT`.`id`
```

А в таком запросе всегда сначала будет прочитана первая таблица:

```
SELECT * FROM `firstT` STRAIGHT_JOIN `secondT`
ON `firstT`.`id` = `secondT`.`id`
```



**В руководстве сказано...**

Использование STRAIGHT\_JOIN не даёт эффекта с таблицами, которые оптимизатор считает таблицами вида «[const](#)» или «[system](#)». В результате обращения к таким таблицам получается один ряд, данные извлекаются на стадии оптимизации запроса, а ссылки на поля такой таблицы заменяются конкретными значениями перед выполнением JOIN. **НО!** Это исключение не актуально для «NULL-стороны» OUTER JOIN («правой» в LEFT JOIN или «левой» в RIGHT JOIN).

Таблица, полученная в результате такого запроса, будет считаться таблицей типа **const** (если вы вдруг забыли, напоминаем: запросы в SQL возвращают таблицы ☺):

```
SELECT * FROM `tbl_name` WHERE `primary_key`=1;
```

### 6.1.5. SELECT: [SQL\_SMALL\_RESULT] [SQL\_BIG\_RESULT]

```

SELECT
[ALL | DISTINCT | DISTINCTROW ]
[HIGH_PRIORITY]
[STRAIGHT_JOIN]
[[SQL_SMALL_RESULT] [[SQL_BIG_RESULT]] [[SQL_BUFFER_RESULT]]
[SQL_CACHE | SQL_NO_CACHE] [[SQL_CALC_FOUND_ROWS]]
[select_type | select_expr ...]
[FROM table_references
[WHERE where_condition]
[GROUP BY col_name [expr | position]
| ANY | ALL | ROLLUP]
[HAVING where_condition]
[ORDER BY {col_name | expr | position}
| ASC | DESC]
[LIMIT | offset | row_count | row_count OFFSET offset]
[PROCEDURE procedure_name(argument_list)]
[INTO OUTFILE 'file_name'
[CHARACTER SET charset_name]
[ROWS rows]
| INTO DUMPFILE 'file_name'
| INTO var_name [, var_name]]
[FOR UPDATE | LOCK IN SHARE MODE]

```



**В руководстве сказано...**

В случае **SQL\_BIG\_RESULT** будет использована временная таблица на диске и временная таблица с ключами на результатах GROUP BY.

В случае **SQL\_SMALL\_RESULT** MySQL будет выполнять операции с временными таблицами в оперативной памяти вместо «дисковой сортировки».

Проведём небольшое исследование (*на самом деле, это «небольшое» исследование вылилось в почти месяц тестов*). Используем таблицу:

```

CREATE TABLE IF NOT EXISTS `t_small_big`
(
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `name` varchar(255) NOT NULL,
    `money` int(11) NOT NULL,
    PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

```

Добавим в таблицу данные и выполним запросы:

```

SELECT `name`, SUM(`money`)
FROM `t_small_big` GROUP BY `name`

SELECT SQL_SMALL_RESULT `name`, SUM(`money`)
FROM `t_small_big` GROUP BY `name`

SELECT SQL_BIG_RESULT `name`, SUM(`money`)
FROM `t_small_big` GROUP BY `name`

```

Предварительные результаты:

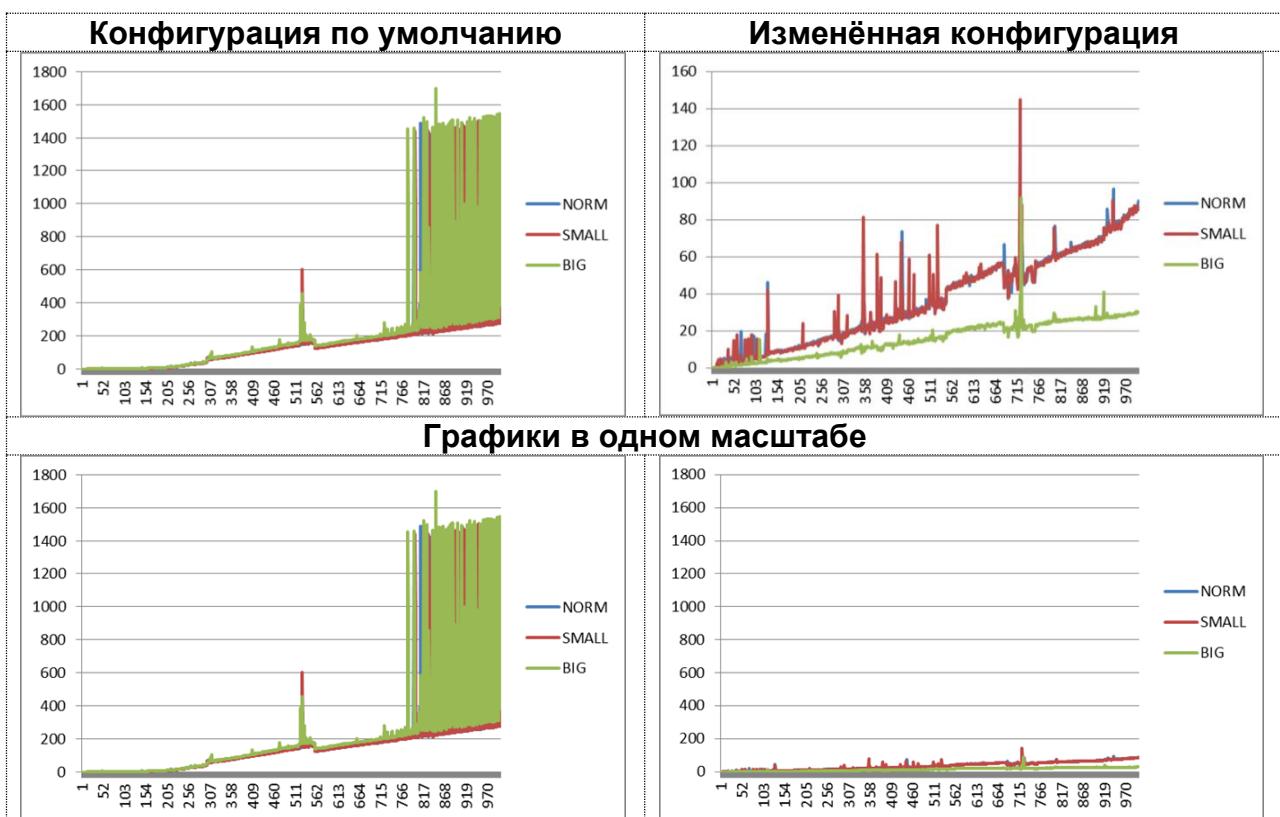
|                 |            | До правки конфигурации |          |                 | После правки конфигурации |          |                 |
|-----------------|------------|------------------------|----------|-----------------|---------------------------|----------|-----------------|
|                 |            | SELECT                 | SMALL    | BIG             | SELECT                    | SMALL    | BIG             |
| 1000 записей    | <b>MIN</b> | 0.00999                | 0.00997  | 0.00433         | 0.00973                   | 0.00995  | 0.00434         |
|                 | <b>MAX</b> | 0.10967                | 0.05353  | 0.02233         | 0.11025                   | 0.05321  | 0.02242         |
|                 | <b>AVG</b> | 0.01142                | 0.01129  | 0.00496         | 0.01023                   | 0.01086  | 0.00455         |
| 1000000 записей | <b>MIN</b> | 30.25444               | 29.37332 | <b>163.7262</b> | 30.66679                  | 30.27623 | <b>10.0718</b>  |
|                 | <b>MAX</b> | 60.28489               | 37.37515 | <b>177.7415</b> | 53.21383                  | 41.92533 | <b>14.16798</b> |
|                 | <b>AVG</b> | 35.36835               | 33.18135 | <b>169.2353</b> | 35.93486                  | 33.30988 | <b>10.81355</b> |

Правка конфигурации включала в себя следующие изменения в my.ini (на мысли о которых навело изучение готового варианта конфигурации my-innodb-heavy-4G.ini):

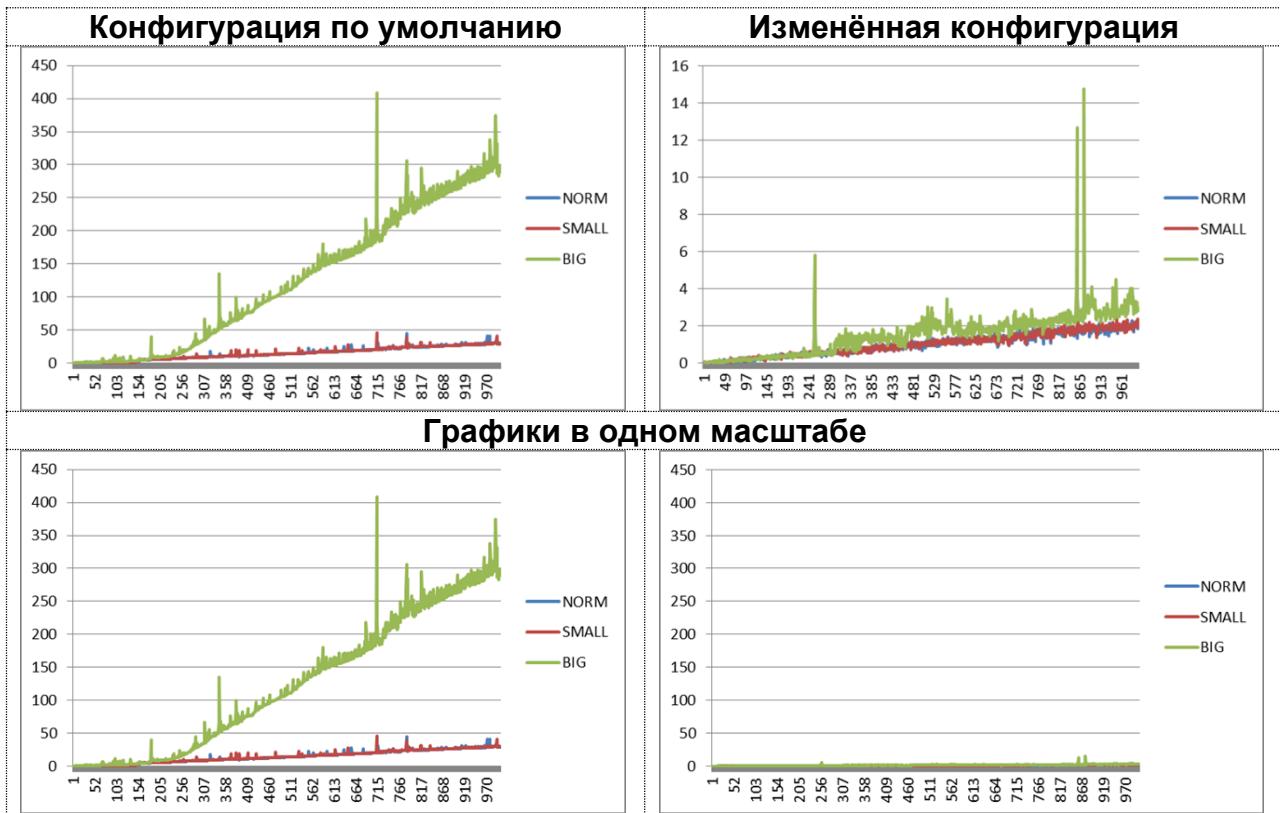
```
# 2 → 20
innodb_additional_mem_pool_size=20M
# 1 → 8
innodb_log_buffer_size=8M
# 8 → 348
innodb_buffer_pool_size=348M
```

После изучения предварительных результатов было решено провести более масштабное исследование.

Сначала тест был проведён при условии, что идентификаторы пользователей, добавляемые к имени (слову 'user') генерировались в диапазоне 1-10000000, тест проводился на объёме данных от 1000 до 1000000 записей с шагом в 1000. Время выполнения запросов приведено в секундах.



Затем тест был проведён при условии, что идентификаторы пользователей, добавляемые к имени (слову 'user') генерировались в диапазоне 1-10, тест проводился на объёме данных от 1000 до 1000000 записей с шагом в 1000. Время выполнения запросов приведено в секундах.



И «финальный штрих».

Идентификаторы пользователей: 1-10000000.

Записей: 1 миллиард.

Конфигурация: **изменена** (см. ранее – как).

Время выполнения запроса («BIG»): **9:29:58.34**

Идентификаторы пользователей: **1-10**.

Записей: 1 миллиард.

Конфигурация: **изменена** (см. ранее – как).

Время выполнения запроса («BIG»): **0:56:3.81**

## 6.1.6. SELECT: [SQL\_BUFFER\_RESULT]

```
SELECT
[ALL | DISTINCT | DISTINCTROW ]
[HIGH_PRIORITY]
[STRAIGHT_JOIN]
[SQL_SMALL_RESULT] [SQL_BIG_RESULT]
[SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
[select_expr [, select_expr ...]]
[FROM table_references]
[WHERE where_condition]
[GROUP BY col_name [expr | position]
[AS] [WITH ROLLUP]]
[HAVING where_condition]
[ORDER BY col_name [expr | position]
[ASC | DESC]
[LIMIT [offset] row_count | row_count OFFSET offset]
[PROCEDURE procedure_name(argument_list)]
[INTO OUTFILE 'file_name'
[CHARACTER SET charset_name]
[export_options]
[INTO DUMPFILE 'file_name'
[INTO var_name [, var_name]]
[FOR UPDATE | LOCK IN SHARE MODE]]]
```

Сообщает MySQL о необходимости поместить результат во временную таблицу, что позволяет быстро снять блокировки (не держать их, пока идёт передача данных клиенту).

**Не работает для подзапросов и UNION.**

## 6.1.7. SELECT: [SQL\_CACHE | SQL\_NO\_CACHE]

```
SELECT
[ALL | DISTINCT | DISTINCTROW ]
[HIGH_PRIORITY]
[STRAIGHT_JOIN]
[SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
[[SQL_CACHE | SQL_NO_CACHE]] [SQL_CALC_FOUND_ROWS]
[select_expr [, select_expr ...]]
[FROM table_references]
[WHERE where_condition]
[GROUP BY col_name [expr | position]
[ASC | DESC] ... [WITH ROLLUP]]
[HAVING where_condition]
[ORDER BY col_name [expr | position]
[ASC | DESC]
[LIMIT [offset] row_count | row_count OFFSET offset]
[PROCEDURE procedure_name(argument_list)]
[INTO OUTFILE 'file_name'
[CHARACTER SET charset_name]
[export_options]
[INTO DUMPFILE 'file_name'
[INTO var_name [, var_name]]
[FOR UPDATE | LOCK IN SHARE MODE]]]
```

Сообщает MySQL о необходимости кэшировать или не кэшировать результаты запроса на выборку.

Заинтересовавшихся нетривиальным поведением MySQL при работе с этими опциями приглашаем к прочтению подробностей здесь:

<http://dev.mysql.com/doc/refman/5.5/en/select.html>

Чтобы при последующем выполнении результата запроса были взяты из кэша, строки запросов должны быть совершенно (до байта!) идентичными.

**Кэш запроса автоматически сбрасывается при ЛЮБОМ изменении (структуре или данных) любой затронутой запросом таблицы.**

Подробнее о поведении кэша в MySQL см. здесь:

<http://dev.mysql.com/doc/refman/5.5/en/query-cache-operation.html>

### 6.1.8. SELECT: [SQL\_CALC\_FOUND\_ROWS]

```

SELECT
[ALL | DISTINCT | DISTINCTROW ]
[HIGH_PRIORITY]
[STRAIGHT_JOIN]
[SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
[SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
select_expr [ , select_expr ... ]
[FROM table_references
[WHERE where_condition]
[GROUP BY col_name [expr | position]
[HAVING where_condition]
[ORDER BY col_name | expr | position]
[AUTO | DESC]
[LIMIT row_count [row_offset OFFSET offset]]
[PROCEDURE procedure_name(argument_list)]
[INTO OUTFILE 'file_name'
[CHARACTER SET charset_name]
[ROWS row_count]
| INTO DUMPFILE 'file_name'
| INTO var_name [, var_name]]
[FOR UPDATE | LOCK IN SHARE MODE]

```

Позволяет выяснить, сколько записей вернул бы запрос, если бы в нём не было конструкции LIMIT.

Для определения общего количества записей используется вызов функции FOUND\_ROWS():

```

SELECT SQL_CALC_FOUND_ROWS * from `tbl` limit 1;
SELECT FOUND_ROWS();

```

Эти запросы нужно выполнять **в рамках одной сессии**, иначе SELECT FOUND\_ROWS() «не поймёт», о каком запросе нужно вернуть данные.

**Внимание! Многие утилиты выполняют отдельные запросы в отдельных сессиях, и потому результат может быть неверным.**

Работает BEPHO:

- Консоль MySQL.
- Расширения PHP для MySQL (если не закрывать соединение).

Работает НЕВЕРНО:

- phpMyAdmin – вообще не показывает результат.
- MySQL WorkBench – показывает неверные данные (в старых версиях).

**Почему это не работает в phpMyAdmin?!?!**

Ответ:

<http://stackoverflow.com/questions/8432819/mysql-found-rows-not-working-from-phpmyadmin>

Процитируем дословно:

*Q: How can I get FOUND\_ROWS() in PMA report properly?*

*A: You can't.*

*PMA uses the FOUND\_ROWS() itself to tell you how many rows were found in your last query, and parses this itself to tell you 'x results found'.*

*Every time you do a simple SELECT, PMA adds a limit and executes this select, and the FOUND\_ROWS afterwards to tell you how many results you would have found in without the LIMIT. This is how PMA offers paging (by default 30 elements), too.*

*Do note also: by default each query you enter in PMA opens a new connection to MySQL, executes the query, and closes the connection. Your next query will not know of the first query, unless you put them all together in the input-textarea, separated by semicolons.*

*If you want to use FOUND\_ROWS(), use another mysql-client. The mysql commandline is fine.*

## Почему это не работает в MySQL Workbench?!?!

Ответ: <http://bugs.mysql.com/bug.php?id=54868>

Процитируем дословно:

*Q: SQL\_CALC\_FOUND\_ROWS and FOUND\_ROWS() does not work in MySQL Workbench SQL Editor, why?*

*A: Fixed in 5.2.26.*

И ответ на частый вопрос ☺.

**Q:** А почему у меня то работает (на некоторых компах), то не работает (на других компах)?!

**A:** Неизвестно. У меня – та же ситуация. Вывод? Используйте консоль MySQL. Там работает всегда ☺.

В контексте определения количества записей часто возникает вопрос о том, что работает быстрее: COUNT(\*), COUNT(1), COUNT('PK'), COUNT('field').

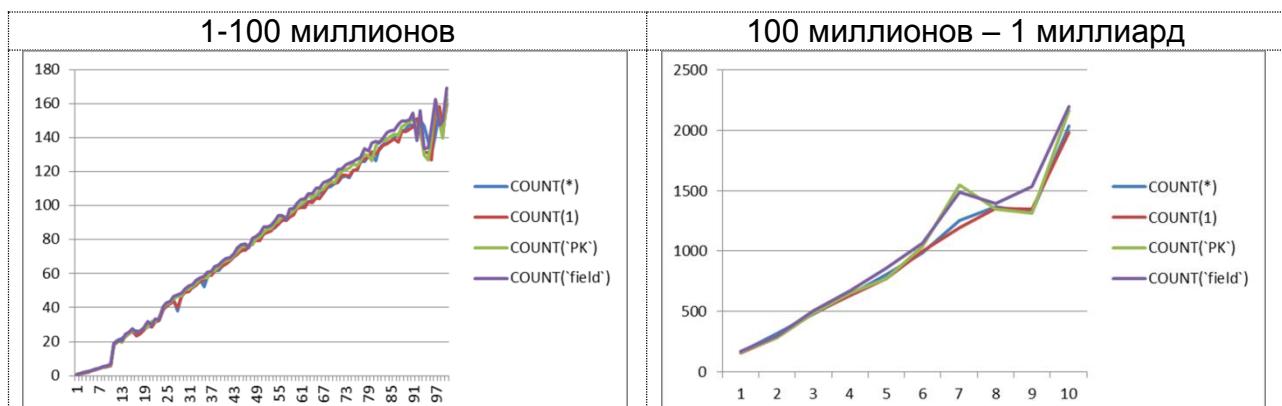
Итак, в ту же таблицу, на которой мы проверяли производительность GROUP BY поместим сначала миллион записей, а потом – миллиард.

Время указано в секундах, проведено 10 итераций теста.

|      |     | COUNT(*)   | COUNT(1)   | COUNT('PK') | COUNT('field') |
|------|-----|------------|------------|-------------|----------------|
| Млн  | MIN | 0.55010    | 0.53930    | 0.53200     | 0.62310        |
|      | MAX | 1.82077    | 0.68506    | 0.74295     | 0.84075        |
|      | AVG | 0.71866    | 0.6088     | 0.62822     | 0.69720        |
| Млрд | MIN | 1463.10896 | 1591.57220 | 1582.18963  | 1573.26027     |
|      | MAX | 1759.67304 | 1756.99959 | 1795.03816  | 1841.69866     |
|      | AVG | 1624.72352 | 1661.70349 | 1691.10235  | 1727.90323     |

Как показывает исследование, резкого отклонения нет. Все три способа работают примерно с одинаковой скоростью.

Дополнительно исследование показывает вот такой рост времени выполнения запросов (записи добавлялись от 1 до 100 миллионов с шагом в 1 миллион, затем записи добавлялись до 1 миллиарда с шагом в 100 миллионов; время указано в секундах):



Можно заметить, что результаты на 1 миллиарде оказались больше (2000+ vs 1700+ секунд), чем в предыдущем тесте с 10-ю итерациями. Скорее всего, сказалась фрагментация ФС.

### 6.1.9. SELECT: select\_expr [, select\_expr ...]

```

SELECT
[ALL | DISTINCT | DISTINCTROW ]
[HIGH_PRIORITY]
[STRAIGHT_JOIN]
[SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
[SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
select_expr [, select_expr ...]

[FROM table_references
[WHERE where_condition]
[GROUP BY col_name [expr | position]
| AVG() | COUNT() | MAX() | MIN() | ROLLUP]
[HAVING where_condition]
[ORDER BY col_name | expr | position]
[ASC | DESC]
[LIMIT row_count | row_offset OFFSET offset]
[PROCEDURE procedure_name(argument_list)]
[INTO OUTFILE 'file_name'
[CHARACTER SET charset_name]
[ROWS row_count]
| INTO DUMPFILE 'file_name'
| INTO var_name [, var_name]]
[FOR UPDATE | LOCK IN SHARE MODE]

```

Основная часть оператора SELECT. Описывает, что именно, следует выбирать.

Множество особенностей select\_expr описано в официальном руководстве здесь: <http://dev.mysql.com/doc/refman/5.5/en/select.html>

#### Примеры использования

##### 1) Выбор всех полей таблицы

| Исходная таблица   | Запрос      | Результат    |              |   |      |    |   |      |    |   |      |    |   |      |   |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |
|--|-------------|--------------|--------------|---|------|----|---|------|----|---|------|----|---|------|---|--------------------------------------|--|-----------|-------------|--------------|---|------|----|---|------|----|---|------|----|---|------|---|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>John</td><td>10</td></tr> <tr> <td>2</td><td>Jack</td><td>20</td></tr> <tr> <td>3</td><td>Alex</td><td>30</td></tr> <tr> <td>4</td><td>John</td><td>5</td></tr> </tbody> </table> | <b>id</b>   | <b>name</b>  | <b>money</b> | 1 | John | 10 | 2 | Jack | 20 | 3 | Alex | 30 | 4 | John | 5 | <code>SELECT * FROM `payment`</code> | <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>John</td><td>10</td></tr> <tr> <td>2</td><td>Jack</td><td>20</td></tr> <tr> <td>3</td><td>Alex</td><td>30</td></tr> <tr> <td>4</td><td>John</td><td>5</td></tr> </tbody> </table> | <b>id</b> | <b>name</b> | <b>money</b> | 1 | John | 10 | 2 | Jack | 20 | 3 | Alex | 30 | 4 | John | 5 |
| <b>id</b>  | <b>name</b> | <b>money</b> |              |   |      |    |   |      |    |   |      |    |   |      |   |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |
| 1  | John        | 10           |              |   |      |    |   |      |    |   |      |    |   |      |   |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |
| 2  | Jack        | 20           |              |   |      |    |   |      |    |   |      |    |   |      |   |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |
| 3  | Alex        | 30           |              |   |      |    |   |      |    |   |      |    |   |      |   |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |
| 4  | John        | 5            |              |   |      |    |   |      |    |   |      |    |   |      |   |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |
| <b>id</b>  | <b>name</b> | <b>money</b> |              |   |      |    |   |      |    |   |      |    |   |      |   |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |
| 1  | John        | 10           |              |   |      |    |   |      |    |   |      |    |   |      |   |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |
| 2  | Jack        | 20           |              |   |      |    |   |      |    |   |      |    |   |      |   |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |
| 3  | Alex        | 30           |              |   |      |    |   |      |    |   |      |    |   |      |   |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |
| 4  | John        | 5            |              |   |      |    |   |      |    |   |      |    |   |      |   |                                      |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |

У запроса вида «SELECT \* ...» есть...

| Преимущества<br>(на малых объёмах данных) | Недостатки<br>(на больших объёмах данных) |
|---|---|
| Не нужно вспоминать имена колонок.        | Получение данных в ненужных объёмах.      |
| Колонки не будут пропущены.               | Легко повесить базу бездумным запросом.   |

##### 2) Выбор отдельных полей таблицы

| Исходная таблица   | Запрос       | Результат    |              |   |      |    |   |      |    |   |      |    |   |      |   |  |  |           |              |   |    |   |    |   |    |   |   |
|--|--------------|--------------|--------------|---|------|----|---|------|----|---|------|----|---|------|---|--|--|-----------|--------------|---|----|---|----|---|----|---|---|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>John</td><td>10</td></tr> <tr> <td>2</td><td>Jack</td><td>20</td></tr> <tr> <td>3</td><td>Alex</td><td>30</td></tr> <tr> <td>4</td><td>John</td><td>5</td></tr> </tbody> </table> | <b>id</b>    | <b>name</b>  | <b>money</b> | 1 | John | 10 | 2 | Jack | 20 | 3 | Alex | 30 | 4 | John | 5 | <code>SELECT `id`, `money` FROM `payment`</code> | <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>10</td></tr> <tr> <td>2</td><td>20</td></tr> <tr> <td>3</td><td>30</td></tr> <tr> <td>4</td><td>5</td></tr> </tbody> </table> | <b>id</b> | <b>money</b> | 1 | 10 | 2 | 20 | 3 | 30 | 4 | 5 |
| <b>id</b>  | <b>name</b>  | <b>money</b> |              |   |      |    |   |      |    |   |      |    |   |      |   |  |  |           |              |   |    |   |    |   |    |   |   |
| 1  | John         | 10           |              |   |      |    |   |      |    |   |      |    |   |      |   |  |  |           |              |   |    |   |    |   |    |   |   |
| 2  | Jack         | 20           |              |   |      |    |   |      |    |   |      |    |   |      |   |  |  |           |              |   |    |   |    |   |    |   |   |
| 3  | Alex         | 30           |              |   |      |    |   |      |    |   |      |    |   |      |   |  |  |           |              |   |    |   |    |   |    |   |   |
| 4  | John         | 5            |              |   |      |    |   |      |    |   |      |    |   |      |   |  |  |           |              |   |    |   |    |   |    |   |   |
| <b>id</b>  | <b>money</b> |              |              |   |      |    |   |      |    |   |      |    |   |      |   |  |  |           |              |   |    |   |    |   |    |   |   |
| 1  | 10           |              |              |   |      |    |   |      |    |   |      |    |   |      |   |  |  |           |              |   |    |   |    |   |    |   |   |
| 2  | 20           |              |              |   |      |    |   |      |    |   |      |    |   |      |   |  |  |           |              |   |    |   |    |   |    |   |   |
| 3  | 30           |              |              |   |      |    |   |      |    |   |      |    |   |      |   |  |  |           |              |   |    |   |    |   |    |   |   |
| 4  | 5            |              |              |   |      |    |   |      |    |   |      |    |   |      |   |  |  |           |              |   |    |   |    |   |    |   |   |

## 3) Выбор полей с переименованием

| Исходная таблица   | Запрос        | Результат    |              |   |      |    |   |      |    |   |      |    |   |      |   |   |  |                    |               |   |    |   |    |   |    |   |   |
|--|---------------|--------------|--------------|---|------|----|---|------|----|---|------|----|---|------|---|---|--|--------------------|---------------|---|----|---|----|---|----|---|---|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>John</td><td>10</td></tr> <tr> <td>2</td><td>Jack</td><td>20</td></tr> <tr> <td>3</td><td>Alex</td><td>30</td></tr> <tr> <td>4</td><td>John</td><td>5</td></tr> </tbody> </table> | <b>id</b>     | <b>name</b>  | <b>money</b> | 1 | John | 10 | 2 | Jack | 20 | 3 | Alex | 30 | 4 | John | 5 | <pre>SELECT `id` AS `transaction`, `money` AS `amount` FROM `payment`</pre> | <table border="1"> <thead> <tr> <th><b>transaction</b></th><th><b>amount</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>10</td></tr> <tr> <td>2</td><td>20</td></tr> <tr> <td>3</td><td>30</td></tr> <tr> <td>4</td><td>5</td></tr> </tbody> </table> | <b>transaction</b> | <b>amount</b> | 1 | 10 | 2 | 20 | 3 | 30 | 4 | 5 |
| <b>id</b>  | <b>name</b>   | <b>money</b> |              |   |      |    |   |      |    |   |      |    |   |      |   |   |  |                    |               |   |    |   |    |   |    |   |   |
| 1  | John          | 10           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |  |                    |               |   |    |   |    |   |    |   |   |
| 2  | Jack          | 20           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |  |                    |               |   |    |   |    |   |    |   |   |
| 3  | Alex          | 30           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |  |                    |               |   |    |   |    |   |    |   |   |
| 4  | John          | 5            |              |   |      |    |   |      |    |   |      |    |   |      |   |   |  |                    |               |   |    |   |    |   |    |   |   |
| <b>transaction</b>   | <b>amount</b> |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |  |                    |               |   |    |   |    |   |    |   |   |
| 1  | 10            |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |  |                    |               |   |    |   |    |   |    |   |   |
| 2  | 20            |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |  |                    |               |   |    |   |    |   |    |   |   |
| 3  | 30            |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |  |                    |               |   |    |   |    |   |    |   |   |
| 4  | 5             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |  |                    |               |   |    |   |    |   |    |   |   |

## 4) Выбор использованием агрегатных функций

| Исходная таблица   | Запрос      | Результат    |              |   |      |    |   |      |    |   |      |    |   |      |   |   |   |            |            |            |            |   |    |         |    |
|--|-------------|--------------|--------------|---|------|----|---|------|----|---|------|----|---|------|---|---|---|------------|------------|------------|------------|---|----|---------|----|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>John</td><td>10</td></tr> <tr> <td>2</td><td>Jack</td><td>20</td></tr> <tr> <td>3</td><td>Alex</td><td>30</td></tr> <tr> <td>4</td><td>John</td><td>5</td></tr> </tbody> </table> | <b>id</b>   | <b>name</b>  | <b>money</b> | 1 | John | 10 | 2 | Jack | 20 | 3 | Alex | 30 | 4 | John | 5 | <pre>SELECT MIN(`money`) as `MIN`, MAX(`money`) as `MAX`, AVG(`money`) as `AVG`, SUM(`money`) as `SUM` FROM `payment`</pre> | <table border="1"> <thead> <tr> <th><b>MIN</b></th><th><b>MAX</b></th><th><b>AVG</b></th><th><b>SUM</b></th></tr> </thead> <tbody> <tr> <td>5</td><td>30</td><td>16.2500</td><td>65</td></tr> </tbody> </table> | <b>MIN</b> | <b>MAX</b> | <b>AVG</b> | <b>SUM</b> | 5 | 30 | 16.2500 | 65 |
| <b>id</b>  | <b>name</b> | <b>money</b> |              |   |      |    |   |      |    |   |      |    |   |      |   |   |   |            |            |            |            |   |    |         |    |
| 1  | John        | 10           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |   |            |            |            |            |   |    |         |    |
| 2  | Jack        | 20           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |   |            |            |            |            |   |    |         |    |
| 3  | Alex        | 30           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |   |            |            |            |            |   |    |         |    |
| 4  | John        | 5            |              |   |      |    |   |      |    |   |      |    |   |      |   |   |   |            |            |            |            |   |    |         |    |
| <b>MIN</b>   | <b>MAX</b>  | <b>AVG</b>   | <b>SUM</b>   |   |      |    |   |      |    |   |      |    |   |      |   |   |   |            |            |            |            |   |    |         |    |
| 5  | 30          | 16.2500      | 65           |   |      |    |   |      |    |   |      |    |   |      |   |   |   |            |            |            |            |   |    |         |    |

## 5) Выбор даты с конвертацией

| Исходная таблица  | Запрос   | Результат               |                     |            |   |                                  |
|---|--|-------------------------|---------------------|------------|---|----------------------------------|
| <table border="1"> <thead> <tr> <th><b>sql_time</b></th><th><b>unix_time</b></th></tr> </thead> <tbody> <tr> <td>2012-10-10 11:43:34</td><td>1349880243</td></tr> </tbody> </table> | <b>sql_time</b>  | <b>unix_time</b>        | 2012-10-10 11:43:34 | 1349880243 | <pre>SELECT FROM_UNIXTIME(`unix_time`, '%Y-%m-%d %H:%i:%s') as `T` FROM `times`</pre> | <p>T<br/>2012-10-10 17:44:03</p> |
| <b>sql_time</b>   | <b>unix_time</b>   |                         |                     |            |   |                                  |
| 2012-10-10 11:43:34   | 1349880243   |                         |                     |            |   |                                  |
|   | <pre>SELECT UNIX_TIMESTAMP(`sql_time`) as `T` FROM `times`</pre> | <p>T<br/>1349858614</p> |                     |            |   |                                  |

6) Выбор значения последнего добавленного автоинкрементированного первично-го ключа

| Исходная таблица   | Запрос      | Результат    |              |   |      |    |   |      |    |   |      |    |   |      |   |   |  |            |   |
|--|-------------|--------------|--------------|---|------|----|---|------|----|---|------|----|---|------|---|---|--|------------|---|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>John</td><td>10</td></tr> <tr> <td>2</td><td>Jack</td><td>20</td></tr> <tr> <td>3</td><td>Alex</td><td>30</td></tr> <tr> <td>4</td><td>John</td><td>5</td></tr> </tbody> </table> | <b>id</b>   | <b>name</b>  | <b>money</b> | 1 | John | 10 | 2 | Jack | 20 | 3 | Alex | 30 | 4 | John | 5 | <p>Выполнять эти два запроса нужно в од-ной сессии (не разрывая соединение с СУБД)!</p> <pre>INSERT INTO `payment` (`id`, `name`, `money`) VALUES (NULL, 'Someone', 999); SELECT LAST_INSERT_ID() as `LID`;</pre> | <table border="1"> <thead> <tr> <th><b>LID</b></th></tr> </thead> <tbody> <tr> <td>5</td></tr> </tbody> </table> | <b>LID</b> | 5 |
| <b>id</b>  | <b>name</b> | <b>money</b> |              |   |      |    |   |      |    |   |      |    |   |      |   |   |  |            |   |
| 1  | John        | 10           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |  |            |   |
| 2  | Jack        | 20           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |  |            |   |
| 3  | Alex        | 30           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |  |            |   |
| 4  | John        | 5            |              |   |      |    |   |      |    |   |      |    |   |      |   |   |  |            |   |
| <b>LID</b>   |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |  |            |   |
| 5  |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |  |            |   |

### ВНИМАНИЕ! ЧАСТАЯ ОШИБКА!

При извлечении LAST\_INSERT\_ID() не надо указывать имя таблицы, или надо указывать LIMIT 1, иначе запрос вернёт столько строк (содержащих верное значение), сколько всего строк в таблице:

| Исходная таблица  | Запрос      | Результат    |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |  |   |            |   |   |   |   |   |   |
|---|-------------|--------------|--------------|---|------|----|---|------|----|---|------|----|---|------|---|---|---------|-----|--|---|------------|---|---|---|---|---|---|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>John</td><td>10</td></tr> <tr> <td>2</td><td>Jack</td><td>20</td></tr> <tr> <td>3</td><td>Alex</td><td>30</td></tr> <tr> <td>4</td><td>John</td><td>5</td></tr> <tr> <td>5</td><td>Someone</td><td>999</td></tr> </tbody> </table> | <b>id</b>   | <b>name</b>  | <b>money</b> | 1 | John | 10 | 2 | Jack | 20 | 3 | Alex | 30 | 4 | John | 5 | 5 | Someone | 999 | <pre>INSERT INTO `payment` (`id`, `name`, `money`) VALUES (NULL, 'Someone', 999); SELECT LAST_INSERT_ID() as `LID` FROM `payment`;</pre> | <table border="1"> <thead> <tr> <th><b>LID</b></th></tr> </thead> <tbody> <tr> <td>6</td></tr> <tr> <td>6</td></tr> <tr> <td>6</td></tr> <tr> <td>6</td></tr> <tr> <td>6</td></tr> <tr> <td>6</td></tr> </tbody> </table> | <b>LID</b> | 6 | 6 | 6 | 6 | 6 | 6 |
| <b>id</b>   | <b>name</b> | <b>money</b> |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |  |   |            |   |   |   |   |   |   |
| 1   | John        | 10           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |  |   |            |   |   |   |   |   |   |
| 2   | Jack        | 20           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |  |   |            |   |   |   |   |   |   |
| 3   | Alex        | 30           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |  |   |            |   |   |   |   |   |   |
| 4   | John        | 5            |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |  |   |            |   |   |   |   |   |   |
| 5   | Someone     | 999          |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |  |   |            |   |   |   |   |   |   |
| <b>LID</b>  |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |  |   |            |   |   |   |   |   |   |
| 6   |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |  |   |            |   |   |   |   |   |   |
| 6   |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |  |   |            |   |   |   |   |   |   |
| 6   |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |  |   |            |   |   |   |   |   |   |
| 6   |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |  |   |            |   |   |   |   |   |   |
| 6   |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |  |   |            |   |   |   |   |   |   |
| 6   |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |  |   |            |   |   |   |   |   |   |

7) Выбор количества записей в таблице

| Исходная таблица   | Запрос      | Результат    |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |              |   |
|--|-------------|--------------|--------------|---|------|----|---|------|----|---|------|----|---|------|---|---|---------|-----|---|---------|-----|---|--|--------------|---|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>John</td><td>10</td></tr> <tr> <td>2</td><td>Jack</td><td>20</td></tr> <tr> <td>3</td><td>Alex</td><td>30</td></tr> <tr> <td>4</td><td>John</td><td>5</td></tr> <tr> <td>5</td><td>Someone</td><td>999</td></tr> <tr> <td>6</td><td>Someone</td><td>999</td></tr> </tbody> </table> | <b>id</b>   | <b>name</b>  | <b>money</b> | 1 | John | 10 | 2 | Jack | 20 | 3 | Alex | 30 | 4 | John | 5 | 5 | Someone | 999 | 6 | Someone | 999 | <pre>SELECT COUNT(*) as `total` FROM `payment`;</pre> | <table border="1"> <thead> <tr> <th><b>total</b></th></tr> </thead> <tbody> <tr> <td>6</td></tr> </tbody> </table> | <b>total</b> | 6 |
| <b>id</b>  | <b>name</b> | <b>money</b> |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |              |   |
| 1  | John        | 10           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |              |   |
| 2  | Jack        | 20           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |              |   |
| 3  | Alex        | 30           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |              |   |
| 4  | John        | 5            |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |              |   |
| 5  | Someone     | 999          |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |              |   |
| 6  | Someone     | 999          |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |              |   |
| <b>total</b>   |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |              |   |
| 6  |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |              |   |

## 8) Выбор с объединением полей таблицы

| Исходная таблица   | Запрос      | Результат    |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                    |           |           |           |          |               |               |
|--|-------------|--------------|--------------|---|------|----|---|------|----|---|------|----|---|------|---|---|---------|-----|---|---------|-----|---|--|--------------------|-----------|-----------|-----------|----------|---------------|---------------|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr><td>1</td><td>John</td><td>10</td></tr> <tr><td>2</td><td>Jack</td><td>20</td></tr> <tr><td>3</td><td>Alex</td><td>30</td></tr> <tr><td>4</td><td>John</td><td>5</td></tr> <tr><td>5</td><td>Someone</td><td>999</td></tr> <tr><td>6</td><td>Someone</td><td>999</td></tr> </tbody> </table> | <b>id</b>   | <b>name</b>  | <b>money</b> | 1 | John | 10 | 2 | Jack | 20 | 3 | Alex | 30 | 4 | John | 5 | 5 | Someone | 999 | 6 | Someone | 999 | <pre>SELECT CONCAT(`id`, ',', `name`, ',', `money`) AS `for_parsing` FROM `payment`</pre> | <table border="1"> <thead> <tr> <th><b>for_parsing</b></th></tr> </thead> <tbody> <tr><td>1;John;10</td></tr> <tr><td>2;Jack;20</td></tr> <tr><td>3;Alex;30</td></tr> <tr><td>4;John;5</td></tr> <tr><td>5;Someone;999</td></tr> <tr><td>6;Someone;999</td></tr> </tbody> </table> | <b>for_parsing</b> | 1;John;10 | 2;Jack;20 | 3;Alex;30 | 4;John;5 | 5;Someone;999 | 6;Someone;999 |
| <b>id</b>  | <b>name</b> | <b>money</b> |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                    |           |           |           |          |               |               |
| 1  | John        | 10           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                    |           |           |           |          |               |               |
| 2  | Jack        | 20           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                    |           |           |           |          |               |               |
| 3  | Alex        | 30           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                    |           |           |           |          |               |               |
| 4  | John        | 5            |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                    |           |           |           |          |               |               |
| 5  | Someone     | 999          |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                    |           |           |           |          |               |               |
| 6  | Someone     | 999          |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                    |           |           |           |          |               |               |
| <b>for_parsing</b>   |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                    |           |           |           |          |               |               |
| 1;John;10  |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                    |           |           |           |          |               |               |
| 2;Jack;20  |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                    |           |           |           |          |               |               |
| 3;Alex;30  |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                    |           |           |           |          |               |               |
| 4;John;5   |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                    |           |           |           |          |               |               |
| 5;Someone;999  |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                    |           |           |           |          |               |               |
| 6;Someone;999  |             |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |                    |           |           |           |          |               |               |

Хороший пример практического применения CONCAT() с habrahabr.ru:

<http://habrahabr.ru/post/153343/>

Получаем список долго выполняемых запросов, в данном случае – SELECT'ы, выполняемые дольше 100 секунд:

```
SELECT CONCAT('KILL ', id, ',') FROM
`information_schema`.`processlist`
WHERE `info` LIKE('SELECT%') AND `time` > 100
INTO OUTFILE '/tmp/killList.sql';
```

Собственно, убиваем их:

```
SOURCE /tmp/killList.sql;
```

## 9) Выбор с объединением строк таблицы

| Исходная таблица  | Запрос      | Результат     |              |             |   |      |     |            |   |      |     |            |   |      |   |            |   |      |   |            |   |      |     |            |   |      |   |            |  |   |          |          |               |        |   |      |        |     |      |        |     |           |
|---|-------------|---------------|--------------|-------------|---|------|-----|------------|---|------|-----|------------|---|------|---|------------|---|------|---|------------|---|------|-----|------------|---|------|---|------------|--|---|----------|----------|---------------|--------|---|------|--------|-----|------|--------|-----|-----------|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th><th><b>date</b></th></tr> </thead> <tbody> <tr><td>1</td><td>Alex</td><td>100</td><td>2012-10-08</td></tr> <tr><td>2</td><td>Alex</td><td>200</td><td>2012-10-24</td></tr> <tr><td>3</td><td>John</td><td>2</td><td>2012-10-02</td></tr> <tr><td>4</td><td>John</td><td>3</td><td>2012-10-23</td></tr> <tr><td>5</td><td>Alex</td><td>300</td><td>2012-07-02</td></tr> <tr><td>6</td><td>John</td><td>4</td><td>2012-03-12</td></tr> </tbody> </table> | <b>id</b>   | <b>name</b>   | <b>money</b> | <b>date</b> | 1 | Alex | 100 | 2012-10-08 | 2 | Alex | 200 | 2012-10-24 | 3 | John | 2 | 2012-10-02 | 4 | John | 3 | 2012-10-23 | 5 | Alex | 300 | 2012-07-02 | 6 | John | 4 | 2012-03-12 | <pre>SELECT EXTRACT(YEAR_MONTH FROM `date`) as `d`, SUM(`money`) as `s`, GROUP_CONCAT(DISTINCT `name` ORDER BY `name` ASC SEPARATOR ';') as `people` from `payment_with_date` group by `d`</pre> | <table border="1"> <thead> <tr> <th><b>d</b></th><th><b>s</b></th><th><b>people</b></th></tr> </thead> <tbody> <tr><td>201203</td><td>4</td><td>John</td></tr> <tr><td>201207</td><td>300</td><td>Alex</td></tr> <tr><td>201210</td><td>305</td><td>Alex;John</td></tr> </tbody> </table> | <b>d</b> | <b>s</b> | <b>people</b> | 201203 | 4 | John | 201207 | 300 | Alex | 201210 | 305 | Alex;John |
| <b>id</b>   | <b>name</b> | <b>money</b>  | <b>date</b>  |             |   |      |     |            |   |      |     |            |   |      |   |            |   |      |   |            |   |      |     |            |   |      |   |            |  |   |          |          |               |        |   |      |        |     |      |        |     |           |
| 1   | Alex        | 100           | 2012-10-08   |             |   |      |     |            |   |      |     |            |   |      |   |            |   |      |   |            |   |      |     |            |   |      |   |            |  |   |          |          |               |        |   |      |        |     |      |        |     |           |
| 2   | Alex        | 200           | 2012-10-24   |             |   |      |     |            |   |      |     |            |   |      |   |            |   |      |   |            |   |      |     |            |   |      |   |            |  |   |          |          |               |        |   |      |        |     |      |        |     |           |
| 3   | John        | 2             | 2012-10-02   |             |   |      |     |            |   |      |     |            |   |      |   |            |   |      |   |            |   |      |     |            |   |      |   |            |  |   |          |          |               |        |   |      |        |     |      |        |     |           |
| 4   | John        | 3             | 2012-10-23   |             |   |      |     |            |   |      |     |            |   |      |   |            |   |      |   |            |   |      |     |            |   |      |   |            |  |   |          |          |               |        |   |      |        |     |      |        |     |           |
| 5   | Alex        | 300           | 2012-07-02   |             |   |      |     |            |   |      |     |            |   |      |   |            |   |      |   |            |   |      |     |            |   |      |   |            |  |   |          |          |               |        |   |      |        |     |      |        |     |           |
| 6   | John        | 4             | 2012-03-12   |             |   |      |     |            |   |      |     |            |   |      |   |            |   |      |   |            |   |      |     |            |   |      |   |            |  |   |          |          |               |        |   |      |        |     |      |        |     |           |
| <b>d</b>  | <b>s</b>    | <b>people</b> |              |             |   |      |     |            |   |      |     |            |   |      |   |            |   |      |   |            |   |      |     |            |   |      |   |            |  |   |          |          |               |        |   |      |        |     |      |        |     |           |
| 201203  | 4           | John          |              |             |   |      |     |            |   |      |     |            |   |      |   |            |   |      |   |            |   |      |     |            |   |      |   |            |  |   |          |          |               |        |   |      |        |     |      |        |     |           |
| 201207  | 300         | Alex          |              |             |   |      |     |            |   |      |     |            |   |      |   |            |   |      |   |            |   |      |     |            |   |      |   |            |  |   |          |          |               |        |   |      |        |     |      |        |     |           |
| 201210  | 305         | Alex;John     |              |             |   |      |     |            |   |      |     |            |   |      |   |            |   |      |   |            |   |      |     |            |   |      |   |            |  |   |          |          |               |        |   |      |        |     |      |        |     |           |

## 10) Выбор с нумерацией строк в выборке

| Исходная таблица   | Запрос      | Результат    |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |   |             |           |      |   |      |   |      |   |
|--|-------------|--------------|--------------|---|------|----|---|------|----|---|------|----|---|------|---|---|---------|-----|---|---------|-----|---|---|-------------|-----------|------|---|------|---|------|---|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr><td>1</td><td>John</td><td>10</td></tr> <tr><td>2</td><td>Jack</td><td>20</td></tr> <tr><td>3</td><td>Alex</td><td>30</td></tr> <tr><td>4</td><td>John</td><td>5</td></tr> <tr><td>5</td><td>Someone</td><td>999</td></tr> <tr><td>6</td><td>Someone</td><td>999</td></tr> </tbody> </table> | <b>id</b>   | <b>name</b>  | <b>money</b> | 1 | John | 10 | 2 | Jack | 20 | 3 | Alex | 30 | 4 | John | 5 | 5 | Someone | 999 | 6 | Someone | 999 | <pre>SET @rownum=0; SELECT `name`, @rownum:="@rownum+1" AS `rn` FROM `payment` WHERE `money`&lt;30;</pre> | <table border="1"> <thead> <tr> <th><b>name</b></th><th><b>rn</b></th></tr> </thead> <tbody> <tr><td>John</td><td>1</td></tr> <tr><td>Jack</td><td>2</td></tr> <tr><td>John</td><td>3</td></tr> </tbody> </table> | <b>name</b> | <b>rn</b> | John | 1 | Jack | 2 | John | 3 |
| <b>id</b>  | <b>name</b> | <b>money</b> |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |   |             |           |      |   |      |   |      |   |
| 1  | John        | 10           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |   |             |           |      |   |      |   |      |   |
| 2  | Jack        | 20           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |   |             |           |      |   |      |   |      |   |
| 3  | Alex        | 30           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |   |             |           |      |   |      |   |      |   |
| 4  | John        | 5            |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |   |             |           |      |   |      |   |      |   |
| 5  | Someone     | 999          |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |   |             |           |      |   |      |   |      |   |
| 6  | Someone     | 999          |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |   |             |           |      |   |      |   |      |   |
| <b>name</b>  | <b>rn</b>   |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |   |             |           |      |   |      |   |      |   |
| John   | 1           |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |   |             |           |      |   |      |   |      |   |
| Jack   | 2           |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |   |             |           |      |   |      |   |      |   |
| John   | 3           |              |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |   |             |           |      |   |      |   |      |   |

## ВНИМАНИЕ! ЧАСТАЯ ОШИБКА!

| Исходная таблица   |             |              | DISTINCT не будет работать, т.к. нумерация даст гарантированную уникальность строк. | Результат |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |  |  |   |             |           |      |   |      |   |      |   |
|--|-------------|--------------|---|-----------|------|----|---|------|----|---|------|----|---|------|---|---|---------|-----|---|---------|-----|--|--|--|---|-------------|-----------|------|---|------|---|------|---|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr><td>1</td><td>John</td><td>10</td></tr> <tr><td>2</td><td>Jack</td><td>20</td></tr> <tr><td>3</td><td>Alex</td><td>30</td></tr> <tr><td>4</td><td>John</td><td>5</td></tr> <tr><td>5</td><td>Someone</td><td>999</td></tr> <tr><td>6</td><td>Someone</td><td>999</td></tr> </tbody> </table> | <b>id</b>   | <b>name</b>  | <b>money</b>  | 1         | John | 10 | 2 | Jack | 20 | 3 | Alex | 30 | 4 | John | 5 | 5 | Someone | 999 | 6 | Someone | 999 |  |  | <pre>SET @rownum=0; SELECT DISTINCT `name`, @rownum:="@rownum+1 AS `rn`" FROM `payment` WHERE `money`&lt;30;</pre> | <table border="1"> <thead> <tr> <th><b>name</b></th><th><b>rn</b></th></tr> </thead> <tbody> <tr><td>John</td><td>1</td></tr> <tr><td>Jack</td><td>2</td></tr> <tr><td>John</td><td>3</td></tr> </tbody> </table> | <b>name</b> | <b>rn</b> | John | 1 | Jack | 2 | John | 3 |
| <b>id</b>  | <b>name</b> | <b>money</b> |   |           |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |  |  |   |             |           |      |   |      |   |      |   |
| 1  | John        | 10           |   |           |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |  |  |   |             |           |      |   |      |   |      |   |
| 2  | Jack        | 20           |   |           |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |  |  |   |             |           |      |   |      |   |      |   |
| 3  | Alex        | 30           |   |           |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |  |  |   |             |           |      |   |      |   |      |   |
| 4  | John        | 5            |   |           |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |  |  |   |             |           |      |   |      |   |      |   |
| 5  | Someone     | 999          |   |           |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |  |  |   |             |           |      |   |      |   |      |   |
| 6  | Someone     | 999          |   |           |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |  |  |   |             |           |      |   |      |   |      |   |
| <b>name</b>  | <b>rn</b>   |              |   |           |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |  |  |   |             |           |      |   |      |   |      |   |
| John   | 1           |              |   |           |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |  |  |   |             |           |      |   |      |   |      |   |
| Jack   | 2           |              |   |           |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |  |  |   |             |           |      |   |      |   |      |   |
| John   | 3           |              |   |           |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |  |  |   |             |           |      |   |      |   |      |   |

## ВНИМАНИЕ! ЧАСТАЯ ОШИБКА!

| Исходная таблица   |             |              | Запрос       | Результат |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |  |  |  |             |           |      |   |
|--|-------------|--------------|--------------|-----------|------|----|---|------|----|---|------|----|---|------|---|---|---------|-----|---|---------|-----|--|--|--|--|-------------|-----------|------|---|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr><td>1</td><td>John</td><td>10</td></tr> <tr><td>2</td><td>Jack</td><td>20</td></tr> <tr><td>3</td><td>Alex</td><td>30</td></tr> <tr><td>4</td><td>John</td><td>5</td></tr> <tr><td>5</td><td>Someone</td><td>999</td></tr> <tr><td>6</td><td>Someone</td><td>999</td></tr> </tbody> </table> | <b>id</b>   | <b>name</b>  | <b>money</b> | 1         | John | 10 | 2 | Jack | 20 | 3 | Alex | 30 | 4 | John | 5 | 5 | Someone | 999 | 6 | Someone | 999 |  |  | <pre>SET @rownum=0; SELECT `name`, @rownum:="@rownum+1 AS `rn`" FROM `payment` HAVING (`rn`&gt;3 AND `rn`&lt;6);</pre> | <table border="1"> <thead> <tr> <th><b>name</b></th><th><b>rn</b></th></tr> </thead> <tbody> <tr><td>John</td><td>6</td></tr> </tbody> </table> <p>Но почему?!</p> | <b>name</b> | <b>rn</b> | John | 6 |
| <b>id</b>  | <b>name</b> | <b>money</b> |              |           |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |  |  |  |             |           |      |   |
| 1  | John        | 10           |              |           |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |  |  |  |             |           |      |   |
| 2  | Jack        | 20           |              |           |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |  |  |  |             |           |      |   |
| 3  | Alex        | 30           |              |           |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |  |  |  |             |           |      |   |
| 4  | John        | 5            |              |           |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |  |  |  |             |           |      |   |
| 5  | Someone     | 999          |              |           |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |  |  |  |             |           |      |   |
| 6  | Someone     | 999          |              |           |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |  |  |  |             |           |      |   |
| <b>name</b>  | <b>rn</b>   |              |              |           |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |  |  |  |             |           |      |   |
| John   | 6           |              |              |           |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |  |  |  |  |             |           |      |   |

Решение 1 (иногда не срабатывает!):

```
SET @rownum=0;
SELECT DISTINCT `name`, @rownum:="@rownum+1 AS `rn`" FROM `payment` HAVING
(`rn`>3 AND `rn`<6);
```

Решение 2 (случаев «не срабатывания» не отмечено):

```
SET @rownum=0;
SELECT `name`, @rownum:="@rownum+1 AS `rn`" FROM `payment` HAVING (`rn`>3 AND
`rn`<6) ORDER BY `rn` ASC;
```

Решение (общая идея): нужно «заставить» MySQL явно сохранить значения для каждой строки «для будущих операций». Иначе где-то оптимизатор берёт значение «не в тот момент».

И всё равно: **НЕ НАДО** полагаться на это поведение в реальных проектах. На текущий момент проверено, что на разных машинах (с небольшими отличиями в версиях MySQL), оно «то работает, то не работает». Даже если ТОЧНО знать, что в конфигурации и т.д. влияет на ситуацию, никто не гарантирует, что на продакшн-сервере настройки однажды не поменяются так, что «всё помрёт».

### 6.1.10. SELECT: FROM table\_references

```

SELECT
[ALL | DISTINCT | DISTINCTROW ]
[HIGH_PRIORITY]
[STRAIGHT_JOIN]
[SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
[SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
[sql_exp | sql_noexp ...]
[FROM table_references
[WHERE where_condition]
[GROUP BY col_name [expr | position]
| [ASC | DESC] [NULLS FIRST | LAST] ROLLUP]
[HAVING where_condition]
[ORDER BY col_name | expr | position]
| [ASC | DESC]
| [LIMIT | offset | row_count | row_offset] [OFFSET offset]
[PROCEDURE procedure_name(argument_list)]
[INTO OUTFILE 'file_name'
| CHARACTER SET charset_name]
| engine_name
| INTO DUMPFILE 'file_name'
| INTO var_name [, var_name]]
[FOR UPDATE | LOCK IN SHARE MODE]

```

Указание таблиц или списка таблиц, из которых надо произвести выборку.

Как правило, выборка производится из одной таблицы.

| Исходная таблица   | Запрос      | Результат    |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
|--|-------------|--------------|--------------|---|------|----|---|------|----|---|------|----|---|------|---|---|---------|-----|---|---------|-----|---|--|-----------|-------------|--------------|---|------|----|---|------|----|---|------|----|---|------|---|---|---------|-----|---|---------|-----|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr><td>1</td><td>John</td><td>10</td></tr> <tr><td>2</td><td>Jack</td><td>20</td></tr> <tr><td>3</td><td>Alex</td><td>30</td></tr> <tr><td>4</td><td>John</td><td>5</td></tr> <tr><td>5</td><td>Someone</td><td>999</td></tr> <tr><td>6</td><td>Someone</td><td>999</td></tr> </tbody> </table> | <b>id</b>   | <b>name</b>  | <b>money</b> | 1 | John | 10 | 2 | Jack | 20 | 3 | Alex | 30 | 4 | John | 5 | 5 | Someone | 999 | 6 | Someone | 999 | <pre> SELECT `id`, `name`, `money` FROM `payment`</pre> | <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr><td>1</td><td>John</td><td>10</td></tr> <tr><td>2</td><td>Jack</td><td>20</td></tr> <tr><td>3</td><td>Alex</td><td>30</td></tr> <tr><td>4</td><td>John</td><td>5</td></tr> <tr><td>5</td><td>Someone</td><td>999</td></tr> <tr><td>6</td><td>Someone</td><td>999</td></tr> </tbody> </table> | <b>id</b> | <b>name</b> | <b>money</b> | 1 | John | 10 | 2 | Jack | 20 | 3 | Alex | 30 | 4 | John | 5 | 5 | Someone | 999 | 6 | Someone | 999 |
| <b>id</b>  | <b>name</b> | <b>money</b> |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| 1  | John        | 10           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| 2  | Jack        | 20           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| 3  | Alex        | 30           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| 4  | John        | 5            |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| 5  | Someone     | 999          |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| 6  | Someone     | 999          |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| <b>id</b>  | <b>name</b> | <b>money</b> |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| 1  | John        | 10           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| 2  | Jack        | 20           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| 3  | Alex        | 30           |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| 4  | John        | 5            |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| 5  | Someone     | 999          |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |
| 6  | Someone     | 999          |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |   |  |           |             |              |   |      |    |   |      |    |   |      |    |   |      |   |   |         |     |   |         |     |

Примеры случаев, когда выборка производится из нескольких таблиц:

- JOIN (самый часты случай).
- UNION (как правило, в случае обращения к т.н. «операционной» и «архивной» таблицам).

Рассмотрим пример обращения к двум таблицам с использованием JOIN:

| Исходные таблицы и схема  | Запрос       | Результат     |               |   |       |   |   |       |   |   |       |   |   |       |   |   |       |      |   |   |               |              |     |       |     |       |      |       |      |       |
|---|--------------|---------------|---------------|---|-------|---|---|-------|---|---|-------|---|---|-------|---|---|-------|------|---|---|---------------|--------------|-----|-------|-----|-------|------|-------|------|-------|
| <table border="1"> <thead> <tr> <th><b>c_id</b></th><th><b>c_inv</b></th><th><b>c_room</b></th></tr> </thead> <tbody> <tr><td>1</td><td>COMP1</td><td>1</td></tr> <tr><td>2</td><td>COMP2</td><td>1</td></tr> <tr><td>3</td><td>COMP3</td><td>2</td></tr> <tr><td>4</td><td>COMP4</td><td>2</td></tr> <tr><td>5</td><td>COMP5</td><td>NULL</td></tr> </tbody> </table><br><p>The diagram shows two tables: 'computers' and 'rooms'. The 'computers' table has columns c_id, c_inv, and c_room. The 'rooms' table has columns r_id and r_name. A green arrow points from the 'computers' table to the 'rooms' table, indicating a join on the c_room column of 'computers' and the r_id column of 'rooms'.</p> | <b>c_id</b>  | <b>c_inv</b>  | <b>c_room</b> | 1 | COMP1 | 1 | 2 | COMP2 | 1 | 3 | COMP3 | 2 | 4 | COMP4 | 2 | 5 | COMP5 | NULL | <pre> SELECT `r_name`, `c_inv` FROM `rooms` JOIN `computers` ON `r_id` = `c_room`</pre> | <table border="1"> <thead> <tr> <th><b>r_name</b></th><th><b>c_inv</b></th></tr> </thead> <tbody> <tr><td>210</td><td>COMP1</td></tr> <tr><td>210</td><td>COMP2</td></tr> <tr><td>213a</td><td>COMP3</td></tr> <tr><td>213a</td><td>COMP4</td></tr> </tbody> </table> | <b>r_name</b> | <b>c_inv</b> | 210 | COMP1 | 210 | COMP2 | 213a | COMP3 | 213a | COMP4 |
| <b>c_id</b>   | <b>c_inv</b> | <b>c_room</b> |               |   |       |   |   |       |   |   |       |   |   |       |   |   |       |      |   |   |               |              |     |       |     |       |      |       |      |       |
| 1   | COMP1        | 1             |               |   |       |   |   |       |   |   |       |   |   |       |   |   |       |      |   |   |               |              |     |       |     |       |      |       |      |       |
| 2   | COMP2        | 1             |               |   |       |   |   |       |   |   |       |   |   |       |   |   |       |      |   |   |               |              |     |       |     |       |      |       |      |       |
| 3   | COMP3        | 2             |               |   |       |   |   |       |   |   |       |   |   |       |   |   |       |      |   |   |               |              |     |       |     |       |      |       |      |       |
| 4   | COMP4        | 2             |               |   |       |   |   |       |   |   |       |   |   |       |   |   |       |      |   |   |               |              |     |       |     |       |      |       |      |       |
| 5   | COMP5        | NULL          |               |   |       |   |   |       |   |   |       |   |   |       |   |   |       |      |   |   |               |              |     |       |     |       |      |       |      |       |
| <b>r_name</b>   | <b>c_inv</b> |               |               |   |       |   |   |       |   |   |       |   |   |       |   |   |       |      |   |   |               |              |     |       |     |       |      |       |      |       |
| 210   | COMP1        |               |               |   |       |   |   |       |   |   |       |   |   |       |   |   |       |      |   |   |               |              |     |       |     |       |      |       |      |       |
| 210   | COMP2        |               |               |   |       |   |   |       |   |   |       |   |   |       |   |   |       |      |   |   |               |              |     |       |     |       |      |       |      |       |
| 213a  | COMP3        |               |               |   |       |   |   |       |   |   |       |   |   |       |   |   |       |      |   |   |               |              |     |       |     |       |      |       |      |       |
| 213a  | COMP4        |               |               |   |       |   |   |       |   |   |       |   |   |       |   |   |       |      |   |   |               |              |     |       |     |       |      |       |      |       |

Рассмотрим пример обращения к двум таблицам с использованием UNION. В операционной таблице хранятся данные только за последний месяц, а всё остальное перенесено в архивную:

| Исходные таблицы и схема   | Запрос      | Результат   |              |              |      |      |            |     |      |      |            |     |   |  |             |      |      |      |
|--|-------------|-------------|--------------|--------------|------|------|------------|-----|------|------|------------|-----|---|--|-------------|------|------|------|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>date</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr> <td>1001</td><td>John</td><td>2012-10-15</td><td>100</td></tr> <tr> <td>1002</td><td>Jack</td><td>2012-10-16</td><td>200</td></tr> </tbody> </table><br> | <b>id</b>   | <b>name</b> | <b>date</b>  | <b>money</b> | 1001 | John | 2012-10-15 | 100 | 1002 | Jack | 2012-10-16 | 200 | <pre>SELECT `name` FROM `payment_operational` UNION DISTINCT SELECT `name` FROM `payment_archive`</pre> | <table border="1"> <thead> <tr> <th><b>name</b></th></tr> </thead> <tbody> <tr> <td>John</td></tr> <tr> <td>Jack</td></tr> <tr> <td>Alex</td></tr> </tbody> </table> | <b>name</b> | John | Jack | Alex |
| <b>id</b>  | <b>name</b> | <b>date</b> | <b>money</b> |              |      |      |            |     |      |      |            |     |   |  |             |      |      |      |
| 1001   | John        | 2012-10-15  | 100          |              |      |      |            |     |      |      |            |     |   |  |             |      |      |      |
| 1002   | Jack        | 2012-10-16  | 200          |              |      |      |            |     |      |      |            |     |   |  |             |      |      |      |
| <b>name</b>  |             |             |              |              |      |      |            |     |      |      |            |     |   |  |             |      |      |      |
| John   |             |             |              |              |      |      |            |     |      |      |            |     |   |  |             |      |      |      |
| Jack   |             |             |              |              |      |      |            |     |      |      |            |     |   |  |             |      |      |      |
| Alex   |             |             |              |              |      |      |            |     |      |      |            |     |   |  |             |      |      |      |

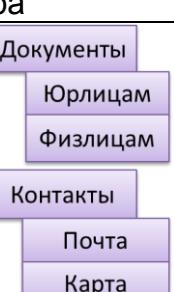
**Q:** Может ли одна и та же таблица несколько раз встречаться в выражении FROM?

**А:** Да. В UNION (хотя это очень редкий случай) и JOIN (проектирование дерева в таблицу, сложное агрегирование).

Пример случая, когда одна и та же таблица встречается в контексте UNION. Например, мы хотим одним запросом выяснить, кто за последний месяц выполнил самый большой и самый маленький платёж:

| Исходная таблица  | Запрос      | Результат    |              |              |      |      |            |     |      |      |            |     |   |   |             |             |              |     |      |     |     |      |     |
|---|-------------|--------------|--------------|--------------|------|------|------------|-----|------|------|------------|-----|---|---|-------------|-------------|--------------|-----|------|-----|-----|------|-----|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>date</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr> <td>1001</td><td>John</td><td>2012-10-15</td><td>100</td></tr> <tr> <td>1002</td><td>Jack</td><td>2012-10-16</td><td>200</td></tr> </tbody> </table> | <b>id</b>   | <b>name</b>  | <b>date</b>  | <b>money</b> | 1001 | John | 2012-10-15 | 100 | 1002 | Jack | 2012-10-16 | 200 | <pre>(SELECT 'MAX' AS `type`, `name`, `money` FROM `payment_operational` ORDER BY `money` DESC LIMIT 1) UNION ALL (SELECT 'MIN' AS `type`, `name`, `money` FROM `payment_operational` ORDER BY `money` ASC LIMIT 1)</pre> | <table border="1"> <thead> <tr> <th><b>type</b></th><th><b>name</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr> <td>MAX</td><td>Jack</td><td>200</td></tr> <tr> <td>MIN</td><td>John</td><td>100</td></tr> </tbody> </table> | <b>type</b> | <b>name</b> | <b>money</b> | MAX | Jack | 200 | MIN | John | 100 |
| <b>id</b>   | <b>name</b> | <b>date</b>  | <b>money</b> |              |      |      |            |     |      |      |            |     |   |   |             |             |              |     |      |     |     |      |     |
| 1001  | John        | 2012-10-15   | 100          |              |      |      |            |     |      |      |            |     |   |   |             |             |              |     |      |     |     |      |     |
| 1002  | Jack        | 2012-10-16   | 200          |              |      |      |            |     |      |      |            |     |   |   |             |             |              |     |      |     |     |      |     |
| <b>type</b>   | <b>name</b> | <b>money</b> |              |              |      |      |            |     |      |      |            |     |   |   |             |             |              |     |      |     |     |      |     |
| MAX   | Jack        | 200          |              |              |      |      |            |     |      |      |            |     |   |   |             |             |              |     |      |     |     |      |     |
| MIN   | John        | 100          |              |              |      |      |            |     |      |      |            |     |   |   |             |             |              |     |      |     |     |      |     |

Пример случая, когда одна и та же таблица встречается в контексте JOIN. Например, нам нужно показать все страницы сайта верхнего уровня с их подстраницами:

| Исходная таблица и структура  | Запрос          | Результат       |               |   |      |           |   |      |          |   |   |         |   |   |          |   |   |       |   |   |       |   |  |               |             |      |           |      |          |           |         |           |          |          |       |          |       |
|---|-----------------|-----------------|---------------|---|------|-----------|---|------|----------|---|---|---------|---|---|----------|---|---|-------|---|---|-------|---|--|---------------|-------------|------|-----------|------|----------|-----------|---------|-----------|----------|----------|-------|----------|-------|
| <table border="1"> <thead> <tr> <th><b>p_id</b></th><th><b>p_parent</b></th><th><b>p_name</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>NULL</td><td>Документы</td></tr> <tr> <td>2</td><td>NULL</td><td>Контакты</td></tr> <tr> <td>3</td><td>1</td><td>Юрлицам</td></tr> <tr> <td>4</td><td>1</td><td>Физлицам</td></tr> <tr> <td>5</td><td>2</td><td>Почта</td></tr> <tr> <td>6</td><td>2</td><td>Карта</td></tr> </tbody> </table><br> | <b>p_id</b>     | <b>p_parent</b> | <b>p_name</b> | 1 | NULL | Документы | 2 | NULL | Контакты | 3 | 1 | Юрлицам | 4 | 1 | Физлицам | 5 | 2 | Почта | 6 | 2 | Карта | <pre>SELECT `level1`.`p_name` AS `parent`, `level2`.`p_name` AS `name` FROM `pages` AS `level1` RIGHT JOIN `pages` AS `level2` ON `level1`.`p_id` = `level2`.`p_parent`</pre> | <table border="1"> <thead> <tr> <th><b>parent</b></th><th><b>name</b></th></tr> </thead> <tbody> <tr> <td>NULL</td><td>Документы</td></tr> <tr> <td>NULL</td><td>Контакты</td></tr> <tr> <td>Документы</td><td>Юрлицам</td></tr> <tr> <td>Документы</td><td>Физлицам</td></tr> <tr> <td>Контакты</td><td>Почта</td></tr> <tr> <td>Контакты</td><td>Карта</td></tr> </tbody> </table> | <b>parent</b> | <b>name</b> | NULL | Документы | NULL | Контакты | Документы | Юрлицам | Документы | Физлицам | Контакты | Почта | Контакты | Карта |
| <b>p_id</b>   | <b>p_parent</b> | <b>p_name</b>   |               |   |      |           |   |      |          |   |   |         |   |   |          |   |   |       |   |   |       |   |  |               |             |      |           |      |          |           |         |           |          |          |       |          |       |
| 1   | NULL            | Документы       |               |   |      |           |   |      |          |   |   |         |   |   |          |   |   |       |   |   |       |   |  |               |             |      |           |      |          |           |         |           |          |          |       |          |       |
| 2   | NULL            | Контакты        |               |   |      |           |   |      |          |   |   |         |   |   |          |   |   |       |   |   |       |   |  |               |             |      |           |      |          |           |         |           |          |          |       |          |       |
| 3   | 1               | Юрлицам         |               |   |      |           |   |      |          |   |   |         |   |   |          |   |   |       |   |   |       |   |  |               |             |      |           |      |          |           |         |           |          |          |       |          |       |
| 4   | 1               | Физлицам        |               |   |      |           |   |      |          |   |   |         |   |   |          |   |   |       |   |   |       |   |  |               |             |      |           |      |          |           |         |           |          |          |       |          |       |
| 5   | 2               | Почта           |               |   |      |           |   |      |          |   |   |         |   |   |          |   |   |       |   |   |       |   |  |               |             |      |           |      |          |           |         |           |          |          |       |          |       |
| 6   | 2               | Карта           |               |   |      |           |   |      |          |   |   |         |   |   |          |   |   |       |   |   |       |   |  |               |             |      |           |      |          |           |         |           |          |          |       |          |       |
| <b>parent</b>   | <b>name</b>     |                 |               |   |      |           |   |      |          |   |   |         |   |   |          |   |   |       |   |   |       |   |  |               |             |      |           |      |          |           |         |           |          |          |       |          |       |
| NULL  | Документы       |                 |               |   |      |           |   |      |          |   |   |         |   |   |          |   |   |       |   |   |       |   |  |               |             |      |           |      |          |           |         |           |          |          |       |          |       |
| NULL  | Контакты        |                 |               |   |      |           |   |      |          |   |   |         |   |   |          |   |   |       |   |   |       |   |  |               |             |      |           |      |          |           |         |           |          |          |       |          |       |
| Документы   | Юрлицам         |                 |               |   |      |           |   |      |          |   |   |         |   |   |          |   |   |       |   |   |       |   |  |               |             |      |           |      |          |           |         |           |          |          |       |          |       |
| Документы   | Физлицам        |                 |               |   |      |           |   |      |          |   |   |         |   |   |          |   |   |       |   |   |       |   |  |               |             |      |           |      |          |           |         |           |          |          |       |          |       |
| Контакты  | Почта           |                 |               |   |      |           |   |      |          |   |   |         |   |   |          |   |   |       |   |   |       |   |  |               |             |      |           |      |          |           |         |           |          |          |       |          |       |
| Контакты  | Карта           |                 |               |   |      |           |   |      |          |   |   |         |   |   |          |   |   |       |   |   |       |   |  |               |             |      |           |      |          |           |         |           |          |          |       |          |       |

И третий пример: когда JOIN и UNION встречаются вместе. Допустим, нам нужно показать все комнаты (включая те, где нет компьютеров) и все компьютеры (даже те, что не поставлены ещё ни в одну комнату):

| Исходные таблицы и схема |       |        | Запрос | Результат |
|--------------------------|-------|--------|--------|-----------|
| c_id                     | c_inv | c_room | r_id   | r_name    |
| 1                        | COMP1 | 1      | 1      | 210       |
| 2                        | COMP2 | 1      | 2      | 213a      |
| 3                        | COMP3 | 2      | 3      | 213b      |
| 4                        | COMP4 | 2      | 4      | 216       |
| 5                        | COMP5 | NULL   |        |           |

|  |   |
|--|---|
|  |   |
| <code>joins.computers</code><br>c_id : int(11)<br>c_inv : varchar(255)<br>c_room : int(11) | <code>joins.rooms</code><br>r_id : int(11)<br>r_name : varchar(255) |

```
(SELECT `r_name`, `c_inv`
FROM `rooms` LEFT OUTER JOIN
`computers` ON
`r_id` = `c_room`)
UNION DISTINCT
(SELECT `r_name`, `c_inv`
FROM `rooms` RIGHT OUTER
JOIN `computers` ON
`r_id` = `c_room`)
```

| r_name | c_inv |
|--------|-------|
| 210    | COMP1 |
| 210    | COMP2 |
| 213a   | COMP3 |
| 213a   | COMP4 |
| 213b   | NULL  |
| 216    | NULL  |
| NULL   | COMP5 |

### 6.1.11. SELECT: [WHERE where\_condition]

```

SELECT
  [ALL | DISTINCT | DISTINCTROW ]
  [HIGH_PRIORITY]
  [STRAIGHT_JOIN]
  [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
  [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
  [index_name | ...]
  [FROM table_references]
  [WHERE where_condition]
  [GROUP BY col_name [expr | position]
    [WITH ROLLUP]]
  [HAVING where_condition]
  [ORDER BY {col_name | expr | position}
    [ASC | DESC]
    [LIMIT {[offset],} row_count | row_count OFFSET offset]]
  [PROCEDURE procedure_name(argument_list)]
  [INTO OUTFILE 'file_name'
    [CHARACTER SET charset_name]
    [options]
    [INTO DUMPFILE 'file_name'
    [INTO var_name [, var_name]]]
  [FOR UPDATE | LOCK IN SHARE MODE]

```

Указание простого или составного условия выборки. В случае использования WHERE все параметры условия должны быть такими, чтобы их можно было вычислить **к моменту применения**. Иначе надо использовать HAVING. Операции с WHERE оптимизируются на уровне СУБД.



#### В руководстве сказано...

Значение **where\_condition** вычисляется как TRUE или FALSE для каждого ряда выборки. С WHERE можно использовать любые функции и операторы MySQL **кроме агрегатных функций**.

См. подробности здесь:

<http://dev.mysql.com/doc/refman/5.5/en/expressions.html>

<http://dev.mysql.com/doc/refman/5.5/en/functions.html>

#### 1) Элементарные примеры применения WHERE

| Исходная таблица  | Запрос  | Результат   |              |              |             |              |            |      |            |      |            |    |   |      |            |    |   |   |           |             |             |              |   |      |            |    |
|---|---|---|--------------|--------------|-------------|--------------|------------|------|------------|------|------------|----|---|------|------------|----|---|---|-----------|-------------|-------------|--------------|---|------|------------|----|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>date</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>John</td><td>2012-07-02</td><td>50</td></tr> <tr> <td>2</td><td>Jack</td><td>2012-07-18</td><td>60</td></tr> <tr> <td>3</td><td>Alex</td><td>2012-08-21</td><td>70</td></tr> </tbody> </table> | <b>id</b>   | <b>name</b>   | <b>date</b>  | <b>money</b> | 1           | John         | 2012-07-02 | 50   | 2          | Jack | 2012-07-18 | 60 | 3 | Alex | 2012-08-21 | 70 | <p>Простое условие:<br/> <b>SELECT * FROM `payment_archive` WHERE `money` &gt; 65</b></p> | <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>date</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr> <td>3</td><td>Alex</td><td>2012-08-21</td><td>70</td></tr> </tbody> </table> | <b>id</b> | <b>name</b> | <b>date</b> | <b>money</b> | 3 | Alex | 2012-08-21 | 70 |
| <b>id</b>   | <b>name</b>   | <b>date</b>   | <b>money</b> |              |             |              |            |      |            |      |            |    |   |      |            |    |   |   |           |             |             |              |   |      |            |    |
| 1   | John  | 2012-07-02  | 50           |              |             |              |            |      |            |      |            |    |   |      |            |    |   |   |           |             |             |              |   |      |            |    |
| 2   | Jack  | 2012-07-18  | 60           |              |             |              |            |      |            |      |            |    |   |      |            |    |   |   |           |             |             |              |   |      |            |    |
| 3   | Alex  | 2012-08-21  | 70           |              |             |              |            |      |            |      |            |    |   |      |            |    |   |   |           |             |             |              |   |      |            |    |
| <b>id</b>   | <b>name</b>   | <b>date</b>   | <b>money</b> |              |             |              |            |      |            |      |            |    |   |      |            |    |   |   |           |             |             |              |   |      |            |    |
| 3   | Alex  | 2012-08-21  | 70           |              |             |              |            |      |            |      |            |    |   |      |            |    |   |   |           |             |             |              |   |      |            |    |
|   | <p>Составное условие:<br/> <b>SELECT * FROM `payment_archive` WHERE `name` = 'John' AND `money` &lt; 60</b></p> | <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>date</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>John</td><td>2012-07-02</td><td>50</td></tr> </tbody> </table> | <b>id</b>    | <b>name</b>  | <b>date</b> | <b>money</b> | 1          | John | 2012-07-02 | 50   |            |    |   |      |            |    |   |   |           |             |             |              |   |      |            |    |
| <b>id</b>   | <b>name</b>   | <b>date</b>   | <b>money</b> |              |             |              |            |      |            |      |            |    |   |      |            |    |   |   |           |             |             |              |   |      |            |    |
| 1   | John  | 2012-07-02  | 50           |              |             |              |            |      |            |      |            |    |   |      |            |    |   |   |           |             |             |              |   |      |            |    |

#### 2) Условие с подзапросом

| Исходная таблица  | Запрос      | Результат   |              |              |   |      |            |    |   |      |            |    |   |      |            |    |   |   |           |             |             |              |   |      |            |    |
|---|-------------|-------------|--------------|--------------|---|------|------------|----|---|------|------------|----|---|------|------------|----|---|---|-----------|-------------|-------------|--------------|---|------|------------|----|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>date</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>John</td><td>2012-07-02</td><td>50</td></tr> <tr> <td>2</td><td>Jack</td><td>2012-07-18</td><td>60</td></tr> <tr> <td>3</td><td>Alex</td><td>2012-08-21</td><td>70</td></tr> </tbody> </table> | <b>id</b>   | <b>name</b> | <b>date</b>  | <b>money</b> | 1 | John | 2012-07-02 | 50 | 2 | Jack | 2012-07-18 | 60 | 3 | Alex | 2012-08-21 | 70 | <pre> SELECT * FROM `payment_archive` WHERE `money` =   (SELECT MAX(`money`)   FROM `payment_archive`) </pre> | <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>date</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr> <td>3</td><td>Alex</td><td>2012-08-21</td><td>70</td></tr> </tbody> </table> | <b>id</b> | <b>name</b> | <b>date</b> | <b>money</b> | 3 | Alex | 2012-08-21 | 70 |
| <b>id</b>   | <b>name</b> | <b>date</b> | <b>money</b> |              |   |      |            |    |   |      |            |    |   |      |            |    |   |   |           |             |             |              |   |      |            |    |
| 1   | John        | 2012-07-02  | 50           |              |   |      |            |    |   |      |            |    |   |      |            |    |   |   |           |             |             |              |   |      |            |    |
| 2   | Jack        | 2012-07-18  | 60           |              |   |      |            |    |   |      |            |    |   |      |            |    |   |   |           |             |             |              |   |      |            |    |
| 3   | Alex        | 2012-08-21  | 70           |              |   |      |            |    |   |      |            |    |   |      |            |    |   |   |           |             |             |              |   |      |            |    |
| <b>id</b>   | <b>name</b> | <b>date</b> | <b>money</b> |              |   |      |            |    |   |      |            |    |   |      |            |    |   |   |           |             |             |              |   |      |            |    |
| 3   | Alex        | 2012-08-21  | 70           |              |   |      |            |    |   |      |            |    |   |      |            |    |   |   |           |             |             |              |   |      |            |    |

Ситуация, в которой нельзя применить WHERE – это выполнение запроса, в котором условие выборки не может быть вычислено на момент своего применения (т.е. оно станет известно только после окончания выполнения запроса).

Для начала добавим в нашу таблицу ещё одну строку и выполним запрос на показ плательщика, СУММА платежей которого больше некоторого числа:

| Исходная таблица  | Запрос   | Результат  |                    |                    |      |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |                                |
|---|--|--|--------------------|--------------------|------|------|------------|----|---|------|------------|----|---|------|------------|----|---|------|------------|-----|---|--------------------------------|
| <table border="1"><thead><tr><th><code>id</code></th><th><code>name</code></th><th><code>date</code></th><th><code>money</code></th></tr></thead><tbody><tr><td>1</td><td>John</td><td>2012-07-02</td><td>50</td></tr><tr><td>2</td><td>Jack</td><td>2012-07-18</td><td>60</td></tr><tr><td>3</td><td>Alex</td><td>2012-08-21</td><td>70</td></tr><tr><td>4</td><td>Alex</td><td>2012-05-07</td><td>500</td></tr></tbody></table> | <code>id</code>  | <code>name</code>  | <code>date</code>  | <code>money</code> | 1    | John | 2012-07-02 | 50 | 2 | Jack | 2012-07-18 | 60 | 3 | Alex | 2012-08-21 | 70 | 4 | Alex | 2012-05-07 | 500 | <pre>SELECT `name`, SUM(`money`) AS `s`<br/>FROM `payment_archive` GROUP BY<br/>`name` WHERE `s`&gt;100</pre> | Ошибка. Запрос<br>не выполнен. |
| <code>id</code>   | <code>name</code>  | <code>date</code>  | <code>money</code> |                    |      |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |                                |
| 1   | John   | 2012-07-02   | 50                 |                    |      |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |                                |
| 2   | Jack   | 2012-07-18   | 60                 |                    |      |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |                                |
| 3   | Alex   | 2012-08-21   | 70                 |                    |      |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |                                |
| 4   | Alex   | 2012-05-07   | 500                |                    |      |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |                                |
|   | <pre>SELECT `name`, SUM(`money`) AS `s`<br/>FROM `payment_archive` GROUP BY<br/>`name` HAVING `s`&gt;100</pre> | <table border="1"><thead><tr><th><code>name</code></th><th><code>s</code></th></tr></thead><tbody><tr><td>Alex</td><td>570</td></tr></tbody></table> | <code>name</code>  | <code>s</code>     | Alex | 570  |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |                                |
| <code>name</code>   | <code>s</code>   |  |                    |                    |      |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |                                |
| Alex  | 570  |  |                    |                    |      |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |                                |

### 6.1.12. SELECT: [GROUP BY ...]

```

SELECT
  [ALL | DISTINCT | DISTINCTROW ]
  [HIGH_PRIORITY]
  [STRAIGHT_JOIN]
  [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
  [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
  [order_by]
  [join_type ...]
  [FROM table_references]
  [WHERE where_condition]
  [GROUP BY {col_name | expr | position}
    [ASC | DESC], ... [WITH ROLLUP]]
  [HAVING where_condition]
  [ORDER BY {col_name | expr | position}
    [ASC | DESC], ...]
  [LIMIT row_count | row_count OFFSET offset]
  [PROCEDURE procedure_name(argument_list)]
  [INTO OUTFILE 'file_name'
    [CHARACTER SET charset_name]
    [options]
    [INTO DUMPFILE 'file_name'
    | INTO var_name [, var_name]]
    [FOR UPDATE | LOCK IN SHARE MODE]]

```

Указание имени (или номера) поля или выражения, по которым следует сгруппировать результаты выборки.



**В руководстве сказано...**

При использовании GROUP BY результат сортируется по тем же полям, по которым производилась группировка. Чтобы избежать этого, используйте GROUP BY `field` ORDER BY NULL;

Для более глубокого понимания того, почему вы получаете те или иные результаты с GROUP BY, см.:

<http://dev.mysql.com/doc/refman/5.5/en/group-by-functions-and-modifiers.html>

#### 1) Группировка по имени поля

Показать сумму платежей по каждому плательщику:

| Исходная таблица |             |             |              | Запрос  | Результат   |          |
|------------------|-------------|-------------|--------------|---|-------------|----------|
| <b>id</b>        | <b>name</b> | <b>date</b> | <b>money</b> |   | <b>name</b> | <b>s</b> |
| 1                | John        | 2012-07-02  | 50           | SELECT `name`, SUM(`money`) AS `s`<br>FROM `payment_archive`<br>GROUP BY `name` | Alex        | 570      |
| 2                | Jack        | 2012-07-18  | 60           |   | Jack        | 60       |
| 3                | Alex        | 2012-08-21  | 70           |   | John        | 50       |
| 4                | Alex        | 2012-05-07  | 500          |   |             |          |

#### 2) Группировка по номеру поля

Несмотря на то, что это работает – **НЕ ДЕЛАЙТЕ ТАК!** Группировка по номерам полей «слетит», как только изменится порядок и/или количество полей в основной части запроса.

| Исходная таблица |             |             |              | Запрос   | Результат   |          |
|------------------|-------------|-------------|--------------|--|-------------|----------|
| <b>id</b>        | <b>name</b> | <b>date</b> | <b>money</b> |  | <b>name</b> | <b>s</b> |
| 1                | John        | 2012-07-02  | 50           | SELECT `name`, SUM(`money`) AS `s`<br>FROM `payment_archive`<br>GROUP BY 1 | Alex        | 570      |
| 2                | Jack        | 2012-07-18  | 60           |  | Jack        | 60       |
| 3                | Alex        | 2012-08-21  | 70           |  | John        | 50       |
| 4                | Alex        | 2012-05-07  | 500          |  |             |          |

### 3) Группировка по выражению

Для рассмотрения серии примеров использования выражений немного доработаем нашу таблицу.

Задача) Показать суммы платежей по каждому месяцу:

| Исходная таблица |             |             |              | Запрос   | Результат |          |
|------------------|-------------|-------------|--------------|--|-----------|----------|
| <b>id</b>        | <b>name</b> | <b>date</b> | <b>money</b> |  | <b>d</b>  | <b>s</b> |
| 1                | Alex        | 2012-08-21  | 70           |  |           |          |
| 2                | Jack        | 2012-07-18  | 60           |  |           |          |
| 3                | John        | 2012-07-02  | 50           |  |           |          |
| 4                | Alex        | 2012-05-17  | 200          | SELECT EXTRACT(YEAR_MONTH FROM `date`) AS `d`, SUM(`money`) AS `s` from `payment_archive` GROUP BY `d` | 201204    | 450      |
| 5                | Alex        | 2012-05-07  | 500          |  | 201205    | 700      |
| 6                | Joe         | 2012-04-27  | 150          |  | 201207    | 110      |
| 7                | Joe         | 2012-04-20  | 150          |  | 201208    | 70       |
| 8                | Joe         | 2012-04-10  | 150          |  |           |          |

Задача) Показать суммы платежей, выполненных каждым плательщиком в каждый месяц (группировка по выражению и полю):

| Исходная таблица |             |             |              | Запрос   | Результат   |          |          |
|------------------|-------------|-------------|--------------|--|-------------|----------|----------|
| <b>id</b>        | <b>name</b> | <b>date</b> | <b>money</b> |  | <b>name</b> | <b>d</b> | <b>s</b> |
| 1                | Alex        | 2012-08-21  | 70           |  | Joe         | 201204   | 450      |
| 2                | Jack        | 2012-07-18  | 60           |  | Alex        | 201205   | 700      |
| 3                | John        | 2012-07-02  | 50           |  | Jack        | 201207   | 60       |
| 4                | Alex        | 2012-05-17  | 200          | SELECT `name`, EXTRACT(YEAR_MONTH FROM `date`) AS `d`, SUM(`money`) AS `s` from `payment_archive` GROUP BY `d`, `name` | John        | 201207   | 50       |
| 5                | Alex        | 2012-05-07  | 500          |  | Alex        | 201208   | 70       |
| 6                | Joe         | 2012-04-27  | 150          |  |             |          |          |
| 7                | Joe         | 2012-04-20  | 150          |  |             |          |          |
| 8                | Joe         | 2012-04-10  | 150          |  |             |          |          |

Немного переделав предыдущий пример, можно показать что выражение в секции GROUP BY можно прописывать явно:

| Исходная таблица |             |             |              | Запрос   | Результат   |          |
|------------------|-------------|-------------|--------------|--|-------------|----------|
| <b>id</b>        | <b>name</b> | <b>date</b> | <b>money</b> |  | <b>name</b> | <b>s</b> |
| 1                | Alex        | 2012-08-21  | 70           |  | Joe         | 450      |
| 2                | Jack        | 2012-07-18  | 60           |  | Alex        | 700      |
| 3                | John        | 2012-07-02  | 50           |  | Jack        | 60       |
| 4                | Alex        | 2012-05-17  | 200          | SELECT `name`, SUM(`money`) AS `s` FROM `payment_archive` GROUP BY EXTRACT(YEAR_MONTH FROM `date`), `name` | John        | 50       |
| 5                | Alex        | 2012-05-07  | 500          |  | Alex        | 70       |
| 6                | Joe         | 2012-04-27  | 150          |  |             |          |
| 7                | Joe         | 2012-04-20  | 150          |  |             |          |
| 8                | Joe         | 2012-04-10  | 150          |  |             |          |

#### 4) Группировка данных из разных строк

Допустим, мы хотим увидеть суммы платежей по месяцам и список пла-тельщиков в виде строки:

| Исходная таблица |             |             |              | Запрос   | Результат |          |              |
|------------------|-------------|-------------|--------------|--|-----------|----------|--------------|
| <b>id</b>        | <b>name</b> | <b>date</b> | <b>money</b> |  | <b>d</b>  | <b>s</b> | <b>names</b> |
| 1                | Alex        | 2012-08-21  | 70           |  |           |          |              |
| 2                | Jack        | 2012-07-18  | 60           |  |           |          |              |
| 3                | John        | 2012-07-02  | 50           |  |           |          |              |
| 4                | Alex        | 2012-05-17  | 200          | <pre>SELECT EXTRACT(YEAR_MONTH FROM `date`) AS `d`, SUM(`money`) AS `s`, GROUP_CONCAT(`name`) AS `names` FROM `payment_archive` GROUP BY `d`</pre> | 201204    | 450      | Joe,Joe,Joe  |
| 5                | Alex        | 2012-05-07  | 500          |  | 201205    | 700      | Alex,Alex    |
| 6                | Joe         | 2012-04-27  | 150          |  | 201207    | 110      | Jack,John    |
| 7                | Joe         | 2012-04-20  | 150          |  | 201208    | 70       | Alex         |
| 8                | Joe         | 2012-04-10  | 150          |  |           |          |              |

Чтобы избежать дублирования имён, вывести их по алфавиту и разделить нужными сепараторами, используем такой запрос (добавим в таблицу ещё один ряд для наглядности):

| Исходная таблица |             |             |              | Запрос  | Результат |          |              |
|------------------|-------------|-------------|--------------|---|-----------|----------|--------------|
| <b>id</b>        | <b>name</b> | <b>date</b> | <b>money</b> |   | <b>d</b>  | <b>s</b> | <b>names</b> |
| 1                | Alex        | 2012-08-21  | 70           |   |           |          |              |
| 2                | Jack        | 2012-07-18  | 60           |   |           |          |              |
| 3                | John        | 2012-07-02  | 50           |   |           |          |              |
| 4                | Alex        | 2012-05-17  | 200          | <pre>SELECT EXTRACT(YEAR_MONTH FROM `date`) AS `d`, SUM(`money`) AS `s`, GROUP_CONCAT(DISTINCT `name` ORDER BY `name` ASC SEPARATOR ';') AS `names` FROM `payment_archive` GROUP BY `d`</pre> | 201204    | 1950     | Daniel;Joe   |
| 5                | Alex        | 2012-05-07  | 500          |   | 201205    | 700      | Alex         |
| 6                | Joe         | 2012-04-27  | 150          |   | 201207    | 110      | Jack;John    |
| 7                | Joe         | 2012-04-20  | 150          |   | 201208    | 70       | Alex         |
| 8                | Joe         | 2012-04-10  | 150          |   |           |          |              |
| 9                | Daniel      | 2012-04-09  | 1500         |   |           |          |              |

Группировка по нескольким полям одной таблицы допустима и не будет нарушением нормализации, например, в таком случае: показать сумму платежей по каждому дню по каждому плательщику.

| Исходная таблица | Запрос | Результат |            |      |      |     |
|------------------|--------|-----------|------------|------|------|-----|
| id               | name   | money     | date       | date | name | sum |
| 1                | Alex   | 100       | 2012-10-08 |      |      |     |
| 2                | Alex   | 200       | 2012-10-08 |      |      |     |
| 3                | Alex   | 1000      | 2012-10-07 |      |      |     |
| 4                | Alex   | 2000      | 2012-10-07 |      |      |     |
| 5                | John   | 10000     | 2012-10-07 |      |      |     |
| 6                | John   | 20000     | 2012-10-07 |      |      |     |

```
SELECT `date`, `name`, SUM(`money`)
AS `sum` FROM `payment_with_date`
GROUP BY `date`, `name`
```

**Q:** Что будет, если в группировке поменять местами поля `date` и `name`?

**А:** Фактически, изменится только последовательность записей. Но ее всегда можно управлять с помощью ORDER BY.

| Исходная таблица | Запрос | Результат |            |      |      |     |
|------------------|--------|-----------|------------|------|------|-----|
| id               | name   | money     | date       | date | name | sum |
| 1                | Alex   | 100       | 2012-10-08 |      |      |     |
| 2                | Alex   | 200       | 2012-10-08 |      |      |     |
| 3                | Alex   | 1000      | 2012-10-07 |      |      |     |
| 4                | Alex   | 2000      | 2012-10-07 |      |      |     |
| 5                | John   | 10000     | 2012-10-07 |      |      |     |
| 6                | John   | 20000     | 2012-10-07 |      |      |     |

```
SELECT `date`, `name`, SUM(`money`)
AS `sum` FROM `payment_with_date`
GROUP BY `date`, `name`
```

```
SELECT `date`, `name`, SUM(`money`)
AS `sum` FROM `payment_with_date`
GROUP BY `name`, `date`
```

**Q:** Что будет, если группировать по полю, в котором есть NULL?

**А:** «Все NULL'ы» будут засчитаны как одинаковые значения.

| Исходная таблица | Запрос | Результат |            |      |     |
|------------------|--------|-----------|------------|------|-----|
| id               | name   | money     | date       | name | sum |
| 1                | Alex   | 100       | 2012-10-08 |      |     |
| 2                | Alex   | 200       | 2012-10-08 |      |     |
| 3                | Alex   | 1000      | 2012-10-07 |      |     |
| 4                | Alex   | 2000      | 2012-10-07 |      |     |
| 5                | John   | 10000     | 2012-10-07 |      |     |
| 6                | John   | 20000     | 2012-10-07 |      |     |
| 7                | NULL   | 5000      | 2012-10-16 |      |     |
| 8                | NULL   | 10000     | 2012-10-16 |      |     |

```
SELECT `name`, SUM(`money`) AS `sum`
FROM `payment_with_date`
GROUP BY `name`
```

### 6.1.13. SELECT: [HAVING where\_condition]

```

SELECT
[ALL | DISTINCT | DISTINCTROW ]
[HIGH_PRIORITY]
[STRAIGHT_JOIN]
[SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
[SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
[NO_WRITE_TO_BINLOG ...]
[FROM table_references
[WHERE where_condition]
[GROUP BY col_name | expr | position]
[ASC | DESC]... [WITH ROLLUP]]
[HAVING where_condition]
[ORDER BY {col_name | expr | position}
[ASC | DESC]... [LIMIT {[offset],}row_count | row_count OFFSET offset]
[PROCEDURE procedure_name(argument_list)]
[INTO OUTFILE 'file_name'
[CHARACTER SET charset_name]
[ROWS rows]
[INTO DUMPFILE 'file_name'
[INTO var_name [, var_name]]
[FOR UPDATE | LOCK IN SHARE MODE]

```

Указание простого или составного условия выборки, параметры которого **не могут быть вычислены на момент применения условия**. Фактически – фильтр, налагаемый на результаты запроса после его выполнения. **НЕ оптимизируется!**



#### В руководстве сказано...

По стандарту SQL HAVING должен ссылаться только на поля, указанные в GROUP BY или в агрегатных функциях. Но MySQL позволяет HAVING ссылаться на поля, перечисленные в SELECT и подзапросах.

В случае, если возникает ситуация, когда MySQL не может определить, идёт ли речь о поле из SELECT или из GROUP BY, HAVING работает с полем из GROUP BY.

В связи с данной цитатой из руководства приведём пример того, как писать запросы **НЕ НАДО** (это работает, но такие запросы тяжело сопровождать):

```
SELECT COUNT(`f1`) AS `f2` FROM `t` GROUP BY `f2` HAVING `f2` = 2;
```

Не пишите так!!!

Когда надо применять HAVING? Вспомним пример «показ плательщика, СУММА платежей которого больше некоторого числа».

| Исходная таблица   | Запрос   | Результат   |              |              |      |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |                             |
|--|--|---|--------------|--------------|------|------|------------|----|---|------|------------|----|---|------|------------|----|---|------|------------|-----|---|-----------------------------|
| <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>date</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>John</td><td>2012-07-02</td><td>50</td></tr> <tr> <td>2</td><td>Jack</td><td>2012-07-18</td><td>60</td></tr> <tr> <td>3</td><td>Alex</td><td>2012-08-21</td><td>70</td></tr> <tr> <td>4</td><td>Alex</td><td>2012-05-07</td><td>500</td></tr> </tbody> </table> | <b>id</b>  | <b>name</b>   | <b>date</b>  | <b>money</b> | 1    | John | 2012-07-02 | 50 | 2 | Jack | 2012-07-18 | 60 | 3 | Alex | 2012-08-21 | 70 | 4 | Alex | 2012-05-07 | 500 | <pre>SELECT `name`, SUM(`money`) AS `s` FROM `payment_archive` GROUP BY `name` WHERE `s`&gt;100</pre> | Ошибка. Запрос не выполнен. |
| <b>id</b>  | <b>name</b>  | <b>date</b>   | <b>money</b> |              |      |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |                             |
| 1  | John   | 2012-07-02  | 50           |              |      |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |                             |
| 2  | Jack   | 2012-07-18  | 60           |              |      |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |                             |
| 3  | Alex   | 2012-08-21  | 70           |              |      |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |                             |
| 4  | Alex   | 2012-05-07  | 500          |              |      |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |                             |
|  | <pre>SELECT `name`, SUM(`money`) AS `s` FROM `payment_archive` GROUP BY `name` HAVING `s`&gt;100</pre> | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th><b>name</b></th><th><b>s</b></th></tr> </thead> <tbody> <tr> <td>Alex</td><td>570</td></tr> </tbody> </table> | <b>name</b>  | <b>s</b>     | Alex | 570  |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |                             |
| <b>name</b>  | <b>s</b>   |   |              |              |      |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |                             |
| Alex   | 570  |   |              |              |      |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |                             |

### 6.1.14. SELECT: [ORDER BY ...]

```

SELECT
[ALL | DISTINCT | DISTINCTROW ]
[HIGH_PRIORITY]
[STRAIGHT_JOIN]
[SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
[SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
[order_by | limit_offset | offset_limit]
[FROM table_references]
[WHERE where_condition]
[GROUP BY col_name | expr | position]
[ASC | DESC | NULLS FIRST | LAST]
[HAVING where_condition]
[ORDER BY {col_name | expr | position}
[ASC | DESC], ...]
[LIMIT row_count | row_count OFFSET offset]
[PROCEDURE procedure_name(argument_list)]
[INTO OUTFILE 'file_name'
[CHARACTER SET charset_name]
[expr_type]
[INTO DUMPFILE 'file_name'
[INTO var_name [, var_name]]
[FOR UPDATE | LOCK IN SHARE MODE]

```

Указание имени или номера поля (поляй) или выражения, по которым надо отсортировать результаты выборки.



**В руководстве сказано...**

Если результаты запроса и подзапросов отсортированы по одному и тому же полю в разных направлениях, приоритет имеет сортировка внешнего запроса.

Таким образом, следующий запрос приведёт к сортировке в направлении DESC.

```
(SELECT ... ORDER BY `f1` ASC) ORDER BY `f1` DESC
```

#### 1) Сортировка по одному полю

| Исходная таблица   | Запрос   | Результат   |              |             |              |             |     |            |       |            |     |            |       |            |      |            |       |            |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |     |            |   |      |     |            |   |      |      |            |
|--|--|---|--------------|-------------|--------------|-------------|-----|------------|-------|------------|-----|------------|-------|------------|------|------------|-------|------------|------|------------|---|------|-------|------------|---|------|-------|------------|---|------|------|------------|---|------|-------|------------|--|--|-----------|-------------|--------------|-------------|---|------|-----|------------|---|------|-----|------------|---|------|------|------------|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th><th><b>date</b></th></tr> </thead> <tbody> <tr><td>1</td><td>Alex</td><td>100</td><td>2012-10-08</td></tr> <tr><td>2</td><td>Alex</td><td>200</td><td>2012-10-08</td></tr> <tr><td>3</td><td>Alex</td><td>1000</td><td>2012-10-07</td></tr> <tr><td>4</td><td>Alex</td><td>2000</td><td>2012-10-07</td></tr> <tr><td>5</td><td>John</td><td>10000</td><td>2012-10-07</td></tr> <tr><td>6</td><td>John</td><td>20000</td><td>2012-10-07</td></tr> <tr><td>7</td><td>NULL</td><td>5000</td><td>2012-10-16</td></tr> <tr><td>8</td><td>NULL</td><td>10000</td><td>2012-10-16</td></tr> </tbody> </table> | <b>id</b>  | <b>name</b>   | <b>money</b> | <b>date</b> | 1            | Alex        | 100 | 2012-10-08 | 2     | Alex       | 200 | 2012-10-08 | 3     | Alex       | 1000 | 2012-10-07 | 4     | Alex       | 2000 | 2012-10-07 | 5 | John | 10000 | 2012-10-07 | 6 | John | 20000 | 2012-10-07 | 7 | NULL | 5000 | 2012-10-16 | 8 | NULL | 10000 | 2012-10-16 | <p>По возрастанию (значение по умолчанию):</p> <pre>SELECT * FROM `payment_with_date` ORDER BY `money` ASC</pre> | <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th><th><b>date</b></th></tr> </thead> <tbody> <tr><td>1</td><td>Alex</td><td>100</td><td>2012-10-08</td></tr> <tr><td>2</td><td>Alex</td><td>200</td><td>2012-10-08</td></tr> <tr><td>3</td><td>Alex</td><td>1000</td><td>2012-10-07</td></tr> </tbody> </table> <p>И т.д.</p> | <b>id</b> | <b>name</b> | <b>money</b> | <b>date</b> | 1 | Alex | 100 | 2012-10-08 | 2 | Alex | 200 | 2012-10-08 | 3 | Alex | 1000 | 2012-10-07 |
| <b>id</b>  | <b>name</b>  | <b>money</b>  | <b>date</b>  |             |              |             |     |            |       |            |     |            |       |            |      |            |       |            |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |     |            |   |      |     |            |   |      |      |            |
| 1  | Alex   | 100   | 2012-10-08   |             |              |             |     |            |       |            |     |            |       |            |      |            |       |            |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |     |            |   |      |     |            |   |      |      |            |
| 2  | Alex   | 200   | 2012-10-08   |             |              |             |     |            |       |            |     |            |       |            |      |            |       |            |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |     |            |   |      |     |            |   |      |      |            |
| 3  | Alex   | 1000  | 2012-10-07   |             |              |             |     |            |       |            |     |            |       |            |      |            |       |            |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |     |            |   |      |     |            |   |      |      |            |
| 4  | Alex   | 2000  | 2012-10-07   |             |              |             |     |            |       |            |     |            |       |            |      |            |       |            |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |     |            |   |      |     |            |   |      |      |            |
| 5  | John   | 10000   | 2012-10-07   |             |              |             |     |            |       |            |     |            |       |            |      |            |       |            |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |     |            |   |      |     |            |   |      |      |            |
| 6  | John   | 20000   | 2012-10-07   |             |              |             |     |            |       |            |     |            |       |            |      |            |       |            |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |     |            |   |      |     |            |   |      |      |            |
| 7  | NULL   | 5000  | 2012-10-16   |             |              |             |     |            |       |            |     |            |       |            |      |            |       |            |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |     |            |   |      |     |            |   |      |      |            |
| 8  | NULL   | 10000   | 2012-10-16   |             |              |             |     |            |       |            |     |            |       |            |      |            |       |            |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |     |            |   |      |     |            |   |      |      |            |
| <b>id</b>  | <b>name</b>  | <b>money</b>  | <b>date</b>  |             |              |             |     |            |       |            |     |            |       |            |      |            |       |            |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |     |            |   |      |     |            |   |      |      |            |
| 1  | Alex   | 100   | 2012-10-08   |             |              |             |     |            |       |            |     |            |       |            |      |            |       |            |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |     |            |   |      |     |            |   |      |      |            |
| 2  | Alex   | 200   | 2012-10-08   |             |              |             |     |            |       |            |     |            |       |            |      |            |       |            |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |     |            |   |      |     |            |   |      |      |            |
| 3  | Alex   | 1000  | 2012-10-07   |             |              |             |     |            |       |            |     |            |       |            |      |            |       |            |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |     |            |   |      |     |            |   |      |      |            |
|  | <p>По убыванию:</p> <pre>SELECT * FROM `payment_with_date` ORDER BY `money` DESC</pre> | <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th><th><b>date</b></th></tr> </thead> <tbody> <tr><td>6</td><td>John</td><td>20000</td><td>2012-10-07</td></tr> <tr><td>5</td><td>John</td><td>10000</td><td>2012-10-07</td></tr> <tr><td>8</td><td>NULL</td><td>10000</td><td>2012-10-16</td></tr> </tbody> </table> <p>И т.д.</p> | <b>id</b>    | <b>name</b> | <b>money</b> | <b>date</b> | 6   | John       | 20000 | 2012-10-07 | 5   | John       | 10000 | 2012-10-07 | 8    | NULL       | 10000 | 2012-10-16 |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |     |            |   |      |     |            |   |      |      |            |
| <b>id</b>  | <b>name</b>  | <b>money</b>  | <b>date</b>  |             |              |             |     |            |       |            |     |            |       |            |      |            |       |            |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |     |            |   |      |     |            |   |      |      |            |
| 6  | John   | 20000   | 2012-10-07   |             |              |             |     |            |       |            |     |            |       |            |      |            |       |            |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |     |            |   |      |     |            |   |      |      |            |
| 5  | John   | 10000   | 2012-10-07   |             |              |             |     |            |       |            |     |            |       |            |      |            |       |            |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |     |            |   |      |     |            |   |      |      |            |
| 8  | NULL   | 10000   | 2012-10-16   |             |              |             |     |            |       |            |     |            |       |            |      |            |       |            |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |     |            |   |      |     |            |   |      |      |            |

#### 2) Сортировка по нескольким полям в разных направлениях

| Исходная таблица   | Запрос      | Результат    |              |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |       |            |   |      |      |            |   |      |      |            |   |      |      |            |   |      |     |            |   |      |     |            |   |      |       |            |   |      |       |            |
|--|-------------|--------------|--------------|-------------|---|------|-----|------------|---|------|-----|------------|---|------|------|------------|---|------|------|------------|---|------|-------|------------|---|------|-------|------------|---|------|------|------------|---|------|-------|------------|--|--|-----------|-------------|--------------|-------------|---|------|-------|------------|---|------|------|------------|---|------|------|------------|---|------|------|------------|---|------|-----|------------|---|------|-----|------------|---|------|-------|------------|---|------|-------|------------|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th><th><b>date</b></th></tr> </thead> <tbody> <tr><td>1</td><td>Alex</td><td>100</td><td>2012-10-08</td></tr> <tr><td>2</td><td>Alex</td><td>200</td><td>2012-10-08</td></tr> <tr><td>3</td><td>Alex</td><td>1000</td><td>2012-10-07</td></tr> <tr><td>4</td><td>Alex</td><td>2000</td><td>2012-10-07</td></tr> <tr><td>5</td><td>John</td><td>10000</td><td>2012-10-07</td></tr> <tr><td>6</td><td>John</td><td>20000</td><td>2012-10-07</td></tr> <tr><td>7</td><td>NULL</td><td>5000</td><td>2012-10-16</td></tr> <tr><td>8</td><td>NULL</td><td>10000</td><td>2012-10-16</td></tr> </tbody> </table> | <b>id</b>   | <b>name</b>  | <b>money</b> | <b>date</b> | 1 | Alex | 100 | 2012-10-08 | 2 | Alex | 200 | 2012-10-08 | 3 | Alex | 1000 | 2012-10-07 | 4 | Alex | 2000 | 2012-10-07 | 5 | John | 10000 | 2012-10-07 | 6 | John | 20000 | 2012-10-07 | 7 | NULL | 5000 | 2012-10-16 | 8 | NULL | 10000 | 2012-10-16 | <pre>SELECT * FROM `payment_with_date` ORDER BY `name` ASC, `money` DESC</pre> | <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th><th><b>date</b></th></tr> </thead> <tbody> <tr><td>8</td><td>NULL</td><td>10000</td><td>2012-10-16</td></tr> <tr><td>7</td><td>NULL</td><td>5000</td><td>2012-10-16</td></tr> <tr><td>4</td><td>Alex</td><td>2000</td><td>2012-10-07</td></tr> <tr><td>3</td><td>Alex</td><td>1000</td><td>2012-10-07</td></tr> <tr><td>2</td><td>Alex</td><td>200</td><td>2012-10-08</td></tr> <tr><td>1</td><td>Alex</td><td>100</td><td>2012-10-08</td></tr> <tr><td>6</td><td>John</td><td>20000</td><td>2012-10-07</td></tr> <tr><td>5</td><td>John</td><td>10000</td><td>2012-10-07</td></tr> </tbody> </table> | <b>id</b> | <b>name</b> | <b>money</b> | <b>date</b> | 8 | NULL | 10000 | 2012-10-16 | 7 | NULL | 5000 | 2012-10-16 | 4 | Alex | 2000 | 2012-10-07 | 3 | Alex | 1000 | 2012-10-07 | 2 | Alex | 200 | 2012-10-08 | 1 | Alex | 100 | 2012-10-08 | 6 | John | 20000 | 2012-10-07 | 5 | John | 10000 | 2012-10-07 |
| <b>id</b>  | <b>name</b> | <b>money</b> | <b>date</b>  |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |       |            |   |      |      |            |   |      |      |            |   |      |      |            |   |      |     |            |   |      |     |            |   |      |       |            |   |      |       |            |
| 1  | Alex        | 100          | 2012-10-08   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |       |            |   |      |      |            |   |      |      |            |   |      |      |            |   |      |     |            |   |      |     |            |   |      |       |            |   |      |       |            |
| 2  | Alex        | 200          | 2012-10-08   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |       |            |   |      |      |            |   |      |      |            |   |      |      |            |   |      |     |            |   |      |     |            |   |      |       |            |   |      |       |            |
| 3  | Alex        | 1000         | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |       |            |   |      |      |            |   |      |      |            |   |      |      |            |   |      |     |            |   |      |     |            |   |      |       |            |   |      |       |            |
| 4  | Alex        | 2000         | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |       |            |   |      |      |            |   |      |      |            |   |      |      |            |   |      |     |            |   |      |     |            |   |      |       |            |   |      |       |            |
| 5  | John        | 10000        | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |       |            |   |      |      |            |   |      |      |            |   |      |      |            |   |      |     |            |   |      |     |            |   |      |       |            |   |      |       |            |
| 6  | John        | 20000        | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |       |            |   |      |      |            |   |      |      |            |   |      |      |            |   |      |     |            |   |      |     |            |   |      |       |            |   |      |       |            |
| 7  | NULL        | 5000         | 2012-10-16   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |       |            |   |      |      |            |   |      |      |            |   |      |      |            |   |      |     |            |   |      |     |            |   |      |       |            |   |      |       |            |
| 8  | NULL        | 10000        | 2012-10-16   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |       |            |   |      |      |            |   |      |      |            |   |      |      |            |   |      |     |            |   |      |     |            |   |      |       |            |   |      |       |            |
| <b>id</b>  | <b>name</b> | <b>money</b> | <b>date</b>  |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |       |            |   |      |      |            |   |      |      |            |   |      |      |            |   |      |     |            |   |      |     |            |   |      |       |            |   |      |       |            |
| 8  | NULL        | 10000        | 2012-10-16   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |       |            |   |      |      |            |   |      |      |            |   |      |      |            |   |      |     |            |   |      |     |            |   |      |       |            |   |      |       |            |
| 7  | NULL        | 5000         | 2012-10-16   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |       |            |   |      |      |            |   |      |      |            |   |      |      |            |   |      |     |            |   |      |     |            |   |      |       |            |   |      |       |            |
| 4  | Alex        | 2000         | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |       |            |   |      |      |            |   |      |      |            |   |      |      |            |   |      |     |            |   |      |     |            |   |      |       |            |   |      |       |            |
| 3  | Alex        | 1000         | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |       |            |   |      |      |            |   |      |      |            |   |      |      |            |   |      |     |            |   |      |     |            |   |      |       |            |   |      |       |            |
| 2  | Alex        | 200          | 2012-10-08   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |       |            |   |      |      |            |   |      |      |            |   |      |      |            |   |      |     |            |   |      |     |            |   |      |       |            |   |      |       |            |
| 1  | Alex        | 100          | 2012-10-08   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |       |            |   |      |      |            |   |      |      |            |   |      |      |            |   |      |     |            |   |      |     |            |   |      |       |            |   |      |       |            |
| 6  | John        | 20000        | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |       |            |   |      |      |            |   |      |      |            |   |      |      |            |   |      |     |            |   |      |     |            |   |      |       |            |   |      |       |            |
| 5  | John        | 10000        | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |       |            |   |      |      |            |   |      |      |            |   |      |      |            |   |      |     |            |   |      |     |            |   |      |       |            |   |      |       |            |

### 3) Сортировка по выражению

Отсортируем платежи по дню их выполнения (год и месяц не учитывается):

| Исходная таблица |             |              |             | Запрос |  |  |  | Результат |             |              |             |
|------------------|-------------|--------------|-------------|--------|--|--|--|-----------|-------------|--------------|-------------|
| <b>id</b>        | <b>name</b> | <b>money</b> | <b>date</b> |        |  |  |  | <b>id</b> | <b>name</b> | <b>money</b> | <b>date</b> |
| 1                | Alex        | 100          | 2012-10-08  |        |  |  |  | 8         | NULL        | 10000        | 2012-10-16  |
| 2                | Alex        | 200          | 2012-10-08  |        |  |  |  | 7         | NULL        | 5000         | 2012-10-16  |
| 3                | Alex        | 1000         | 2012-10-07  |        |  |  |  | 4         | Alex        | 2000         | 2012-10-07  |
| 4                | Alex        | 2000         | 2012-10-07  |        |  |  |  | 3         | Alex        | 1000         | 2012-10-07  |
| 5                | John        | 10000        | 2012-10-07  |        |  |  |  | 2         | Alex        | 200          | 2012-10-08  |
| 6                | John        | 20000        | 2012-10-07  |        |  |  |  | 1         | Alex        | 100          | 2012-10-08  |
| 7                | NULL        | 5000         | 2012-10-16  |        |  |  |  | 6         | John        | 20000        | 2012-10-07  |
| 8                | NULL        | 10000        | 2012-10-16  |        |  |  |  | 5         | John        | 10000        | 2012-10-07  |

**ВНИМАНИЕ!** Помните, что сортировка по выражению может очень сильно снизить производительность, т.к. MySQL (пока?) не умеет строить индексы на выражениях.

### 4) Сортировка по номеру поля

```
SELECT * FROM `payment_with_date` ORDER BY 3 DESC
```

Несмотря на то, что это работает – **НЕ ДЕЛАЙТЕ ТАК!** Сортировка по номерам полей «слетит», как только изменится порядок и/или количество полей в таблице.

### 5) Выбор случайной записи из таблицы

| Исходная таблица |             |              |             | Запрос |  |  |  | Результат |             |              |             |
|------------------|-------------|--------------|-------------|--------|--|--|--|-----------|-------------|--------------|-------------|
| <b>id</b>        | <b>name</b> | <b>money</b> | <b>date</b> |        |  |  |  | <b>id</b> | <b>name</b> | <b>money</b> | <b>date</b> |
| 1                | Alex        | 100          | 2012-10-08  |        |  |  |  | 7         | NULL        | 5000         | 2012-10-16  |
| 2                | Alex        | 200          | 2012-10-08  |        |  |  |  |           |             |              |             |
| 3                | Alex        | 1000         | 2012-10-07  |        |  |  |  |           |             |              |             |
| 4                | Alex        | 2000         | 2012-10-07  |        |  |  |  |           |             |              |             |
| 5                | John        | 10000        | 2012-10-07  |        |  |  |  |           |             |              |             |
| 6                | John        | 20000        | 2012-10-07  |        |  |  |  |           |             |              |             |
| 7                | NULL        | 5000         | 2012-10-16  |        |  |  |  |           |             |              |             |
| 8                | NULL        | 10000        | 2012-10-16  |        |  |  |  |           |             |              |             |

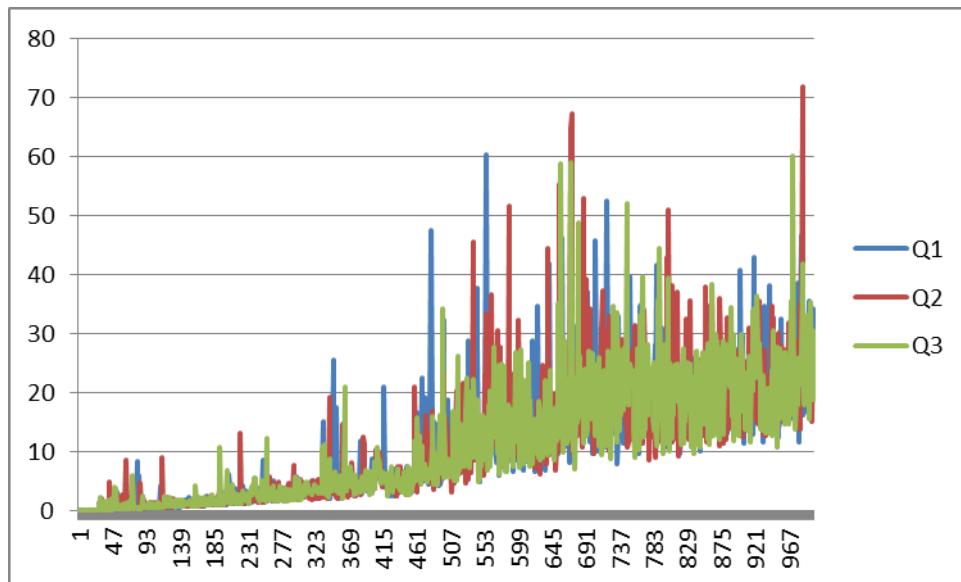
Такое решение может привести к проблемам с производительностью на больших объемах данных.

О том, как выбрать из таблицы некоторое количество случайных уникальных записей, читайте здесь: [http://svyatoslav.biz/engines/random\\_records/](http://svyatoslav.biz/engines/random_records/).

Проверим, как быстро работает такой запрос:

```
SELECT * FROM `payment_with_date` ORDER BY RAND() LIMIT 1
```

Будем добавлять в таблицу (ту же, что и при тестировании производительности GROUP BY) по тысяче записей и трижды выполнять запрос на выборку случайной записи (время указано в секундах).



Итак, с увеличением количества записей скорость работы запроса ощутимо падает.

### 6.1.15. SELECT: [LIMIT ...]

```

SELECT
[ALL | DISTINCT | DISTINCTROW ]
[HIGH_PRIORITY]
[STRAIGHT_JOIN]
[SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
[SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
select_expr [, select_expr ...]
[PARTITION partition_name]
[WHERE where_condition]
[GROUP BY (col_name | expr | position)
[ASC | DESC], ... [WITH ROLLUP]]
[HAVING having_condition]
[ORDER BY (col_name | expr | position)
[ASC | DESC], ...]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
[PROCEDURE procedure_name(argument_list)]
[INTO OUTFILE 'file_name'
[CHARACTER SET charset_name]
export_options]
[INTO DUMPFILE 'file_name'
[INTO var_name [, var_name]]
[FOR UPDATE | LOCK IN SHARE MODE]]

```

Указание количества (и смещения) рядов выборки, которые нужно вернуть.

1) Вернуть первые N рядов выборки

| Исходная таблица   | Запрос      | Результат    |              |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |   |           |             |   |      |
|--|-------------|--------------|--------------|-------------|---|------|-----|------------|---|------|-----|------------|---|------|------|------------|---|------|------|------------|---|------|-------|------------|---|------|-------|------------|---|------|------|------------|---|------|-------|------------|---|---|-----------|-------------|---|------|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th><th><b>date</b></th></tr> </thead> <tbody> <tr><td>1</td><td>Alex</td><td>100</td><td>2012-10-08</td></tr> <tr><td>2</td><td>Alex</td><td>200</td><td>2012-10-08</td></tr> <tr><td>3</td><td>Alex</td><td>1000</td><td>2012-10-07</td></tr> <tr><td>4</td><td>Alex</td><td>2000</td><td>2012-10-07</td></tr> <tr><td>5</td><td>John</td><td>10000</td><td>2012-10-07</td></tr> <tr><td>6</td><td>John</td><td>20000</td><td>2012-10-07</td></tr> <tr><td>7</td><td>NULL</td><td>5000</td><td>2012-10-16</td></tr> <tr><td>8</td><td>NULL</td><td>10000</td><td>2012-10-16</td></tr> </tbody> </table> | <b>id</b>   | <b>name</b>  | <b>money</b> | <b>date</b> | 1 | Alex | 100 | 2012-10-08 | 2 | Alex | 200 | 2012-10-08 | 3 | Alex | 1000 | 2012-10-07 | 4 | Alex | 2000 | 2012-10-07 | 5 | John | 10000 | 2012-10-07 | 6 | John | 20000 | 2012-10-07 | 7 | NULL | 5000 | 2012-10-16 | 8 | NULL | 10000 | 2012-10-16 | <pre> SELECT `id`, `name` FROM `payment_with_date` LIMIT 1 </pre> | <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th></tr> </thead> <tbody> <tr><td>1</td><td>Alex</td></tr> </tbody> </table> | <b>id</b> | <b>name</b> | 1 | Alex |
| <b>id</b>  | <b>name</b> | <b>money</b> | <b>date</b>  |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |   |           |             |   |      |
| 1  | Alex        | 100          | 2012-10-08   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |   |           |             |   |      |
| 2  | Alex        | 200          | 2012-10-08   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |   |           |             |   |      |
| 3  | Alex        | 1000         | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |   |           |             |   |      |
| 4  | Alex        | 2000         | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |   |           |             |   |      |
| 5  | John        | 10000        | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |   |           |             |   |      |
| 6  | John        | 20000        | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |   |           |             |   |      |
| 7  | NULL        | 5000         | 2012-10-16   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |   |           |             |   |      |
| 8  | NULL        | 10000        | 2012-10-16   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |   |           |             |   |      |
| <b>id</b>  | <b>name</b> |              |              |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |   |           |             |   |      |
| 1  | Alex        |              |              |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |   |           |             |   |      |

2) Вернуть N рядов выборки, начиная с M-го

| Исходная таблица   | Запросы эквивалентны | Результат    |              |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |   |      |   |      |
|--|----------------------|--------------|--------------|-------------|---|------|-----|------------|---|------|-----|------------|---|------|------|------------|---|------|------|------------|---|------|-------|------------|---|------|-------|------------|---|------|------|------------|---|------|-------|------------|--|--|-----------|-------------|---|------|---|------|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th><th><b>date</b></th></tr> </thead> <tbody> <tr><td>1</td><td>Alex</td><td>100</td><td>2012-10-08</td></tr> <tr><td>2</td><td>Alex</td><td>200</td><td>2012-10-08</td></tr> <tr><td>3</td><td>Alex</td><td>1000</td><td>2012-10-07</td></tr> <tr><td>4</td><td>Alex</td><td>2000</td><td>2012-10-07</td></tr> <tr><td>5</td><td>John</td><td>10000</td><td>2012-10-07</td></tr> <tr><td>6</td><td>John</td><td>20000</td><td>2012-10-07</td></tr> <tr><td>7</td><td>NULL</td><td>5000</td><td>2012-10-16</td></tr> <tr><td>8</td><td>NULL</td><td>10000</td><td>2012-10-16</td></tr> </tbody> </table> | <b>id</b>            | <b>name</b>  | <b>money</b> | <b>date</b> | 1 | Alex | 100 | 2012-10-08 | 2 | Alex | 200 | 2012-10-08 | 3 | Alex | 1000 | 2012-10-07 | 4 | Alex | 2000 | 2012-10-07 | 5 | John | 10000 | 2012-10-07 | 6 | John | 20000 | 2012-10-07 | 7 | NULL | 5000 | 2012-10-16 | 8 | NULL | 10000 | 2012-10-16 | <pre> SELECT `id`, `name` FROM `payment_with_date` LIMIT 5, 2 </pre><br><pre> SELECT `id`, `name` FROM `payment_with_date` LIMIT 2 OFFSET 5 </pre> | <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th></tr> </thead> <tbody> <tr><td>6</td><td>John</td></tr> <tr><td>7</td><td>NULL</td></tr> </tbody> </table> | <b>id</b> | <b>name</b> | 6 | John | 7 | NULL |
| <b>id</b>  | <b>name</b>          | <b>money</b> | <b>date</b>  |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |   |      |   |      |
| 1  | Alex                 | 100          | 2012-10-08   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |   |      |   |      |
| 2  | Alex                 | 200          | 2012-10-08   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |   |      |   |      |
| 3  | Alex                 | 1000         | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |   |      |   |      |
| 4  | Alex                 | 2000         | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |   |      |   |      |
| 5  | John                 | 10000        | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |   |      |   |      |
| 6  | John                 | 20000        | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |   |      |   |      |
| 7  | NULL                 | 5000         | 2012-10-16   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |   |      |   |      |
| 8  | NULL                 | 10000        | 2012-10-16   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |   |      |   |      |
| <b>id</b>  | <b>name</b>          |              |              |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |   |      |   |      |
| 6  | John                 |              |              |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |   |      |   |      |
| 7  | NULL                 |              |              |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |   |      |   |      |

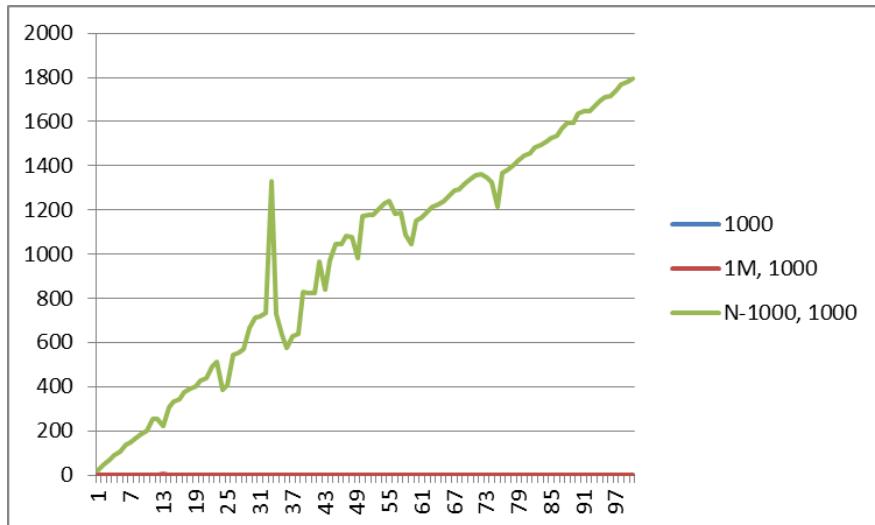
**Q:** Почему «тормозят» запросы вида “SELECT \* FROM `t` LIMIT 1000000000000000000, 5” ?

**A:** Потому, что MySQL выбирает 10000000000000005 рядов, а потом отбрасывает первые 10000000000000000.

Проведём исследование. Во всё ту же таблицу, что мы использовали для теста производительности GROUP BY, будем добавлять по записи до миллиарда с шагом в 10 миллионов, и после каждой итерации будем выполнять три запроса.

|    |  |
|----|--|
| Q1 | SELECT * FROM `t_small_big` limit 1000           |
| Q2 | SELECT * FROM `t_small_big` limit 1000000, 1000  |
| Q3 | SELECT * FROM `t_small_big` limit {N-1000}, 1000 |

Количество записей на текущей



Как показало исследование, MySQL действительно испытывает проблемы с производительностью при использовании большого смещения (OFFSET) в конструкции LIMIT.

Решением этой ситуации может стать «разворачивание» выборки (тогда придётся возвращать несколько ПЕРВЫХ, а не ПОСЛЕДНИХ записей) или наложение дополнительных ограничений на выборку, существенно уменьшающих количество возвращённых рядов.

**Q:** Говорят, в Oracle можно обращаться к номеру ряда выборки? А в MySQL можно?

**A:** Пока – нет. Можно «костылями» эмулировать нумерацию рядов, например – так.

```
SET @rownum=0;
SELECT `name`, @rownum:="@rownum+1 AS `rn` FROM `payment` HAVING (`rn`>3 AND `rn`<6) ORDER BY `rn` ASC;
```

**ВНИМАНИЕ!** Это НЕ ДАСТ никакого ускорения. HAVING всё равно срабатывает после того, как MySQL выберет ВСЕ записи (что ещё хуже, чем в LIMIT, т.к. там будет выбрано только {LIMIT + OFFSET} записей).

**Q:** Как взять последние N рядов выборки?

**A:** Отсортировать в обратном порядке и взять первые N рядов. Например, вот этот запрос вернёт два последних (по `id`) платежа.

| Исходная таблица   | Запрос      | Результат    |              |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |  |           |             |              |             |   |      |       |            |   |      |      |            |
|--|-------------|--------------|--------------|-------------|---|------|-----|------------|---|------|-----|------------|---|------|------|------------|---|------|------|------------|---|------|-------|------------|---|------|-------|------------|---|------|------|------------|---|------|-------|------------|---|--|-----------|-------------|--------------|-------------|---|------|-------|------------|---|------|------|------------|
|  |             |              |              |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |  |           |             |              |             |   |      |       |            |   |      |      |            |
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th><th><b>date</b></th></tr> </thead> <tbody> <tr><td>1</td><td>Alex</td><td>100</td><td>2012-10-08</td></tr> <tr><td>2</td><td>Alex</td><td>200</td><td>2012-10-08</td></tr> <tr><td>3</td><td>Alex</td><td>1000</td><td>2012-10-07</td></tr> <tr><td>4</td><td>Alex</td><td>2000</td><td>2012-10-07</td></tr> <tr><td>5</td><td>John</td><td>10000</td><td>2012-10-07</td></tr> <tr><td>6</td><td>John</td><td>20000</td><td>2012-10-07</td></tr> <tr><td>7</td><td>NULL</td><td>5000</td><td>2012-10-16</td></tr> <tr><td>8</td><td>NULL</td><td>10000</td><td>2012-10-16</td></tr> </tbody> </table> | <b>id</b>   | <b>name</b>  | <b>money</b> | <b>date</b> | 1 | Alex | 100 | 2012-10-08 | 2 | Alex | 200 | 2012-10-08 | 3 | Alex | 1000 | 2012-10-07 | 4 | Alex | 2000 | 2012-10-07 | 5 | John | 10000 | 2012-10-07 | 6 | John | 20000 | 2012-10-07 | 7 | NULL | 5000 | 2012-10-16 | 8 | NULL | 10000 | 2012-10-16 | <pre>SELECT * FROM `payment_with_date` ORDER BY `id` DESC LIMIT 2</pre> | <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th><th><b>date</b></th></tr> </thead> <tbody> <tr><td>8</td><td>NULL</td><td>10000</td><td>2012-10-16</td></tr> <tr><td>7</td><td>NULL</td><td>5000</td><td>2012-10-16</td></tr> </tbody> </table> | <b>id</b> | <b>name</b> | <b>money</b> | <b>date</b> | 8 | NULL | 10000 | 2012-10-16 | 7 | NULL | 5000 | 2012-10-16 |
| <b>id</b>  | <b>name</b> | <b>money</b> | <b>date</b>  |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |  |           |             |              |             |   |      |       |            |   |      |      |            |
| 1  | Alex        | 100          | 2012-10-08   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |  |           |             |              |             |   |      |       |            |   |      |      |            |
| 2  | Alex        | 200          | 2012-10-08   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |  |           |             |              |             |   |      |       |            |   |      |      |            |
| 3  | Alex        | 1000         | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |  |           |             |              |             |   |      |       |            |   |      |      |            |
| 4  | Alex        | 2000         | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |  |           |             |              |             |   |      |       |            |   |      |      |            |
| 5  | John        | 10000        | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |  |           |             |              |             |   |      |       |            |   |      |      |            |
| 6  | John        | 20000        | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |  |           |             |              |             |   |      |       |            |   |      |      |            |
| 7  | NULL        | 5000         | 2012-10-16   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |  |           |             |              |             |   |      |       |            |   |      |      |            |
| 8  | NULL        | 10000        | 2012-10-16   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |  |           |             |              |             |   |      |       |            |   |      |      |            |
| <b>id</b>  | <b>name</b> | <b>money</b> | <b>date</b>  |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |  |           |             |              |             |   |      |       |            |   |      |      |            |
| 8  | NULL        | 10000        | 2012-10-16   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |  |           |             |              |             |   |      |       |            |   |      |      |            |
| 7  | NULL        | 5000         | 2012-10-16   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |   |  |           |             |              |             |   |      |       |            |   |      |      |            |

А как сделать, чтобы теперь здесь значения `id` шли по возрастанию?

| Исходная таблица   | Запрос      | Результат    |              |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |      |            |   |      |       |            |
|--|-------------|--------------|--------------|-------------|---|------|-----|------------|---|------|-----|------------|---|------|------|------------|---|------|------|------------|---|------|-------|------------|---|------|-------|------------|---|------|------|------------|---|------|-------|------------|--|--|-----------|-------------|--------------|-------------|---|------|------|------------|---|------|-------|------------|
|  |             |              |              |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |      |            |   |      |       |            |
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th><th><b>date</b></th></tr> </thead> <tbody> <tr><td>1</td><td>Alex</td><td>100</td><td>2012-10-08</td></tr> <tr><td>2</td><td>Alex</td><td>200</td><td>2012-10-08</td></tr> <tr><td>3</td><td>Alex</td><td>1000</td><td>2012-10-07</td></tr> <tr><td>4</td><td>Alex</td><td>2000</td><td>2012-10-07</td></tr> <tr><td>5</td><td>John</td><td>10000</td><td>2012-10-07</td></tr> <tr><td>6</td><td>John</td><td>20000</td><td>2012-10-07</td></tr> <tr><td>7</td><td>NULL</td><td>5000</td><td>2012-10-16</td></tr> <tr><td>8</td><td>NULL</td><td>10000</td><td>2012-10-16</td></tr> </tbody> </table> | <b>id</b>   | <b>name</b>  | <b>money</b> | <b>date</b> | 1 | Alex | 100 | 2012-10-08 | 2 | Alex | 200 | 2012-10-08 | 3 | Alex | 1000 | 2012-10-07 | 4 | Alex | 2000 | 2012-10-07 | 5 | John | 10000 | 2012-10-07 | 6 | John | 20000 | 2012-10-07 | 7 | NULL | 5000 | 2012-10-16 | 8 | NULL | 10000 | 2012-10-16 | <pre>SELECT * FROM (SELECT * FROM `payment_with_date` ORDER BY `id` DESC LIMIT 2) as `tmp` ORDER BY `id` ASC</pre> | <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th><th><b>date</b></th></tr> </thead> <tbody> <tr><td>7</td><td>NULL</td><td>5000</td><td>2012-10-16</td></tr> <tr><td>8</td><td>NULL</td><td>10000</td><td>2012-10-16</td></tr> </tbody> </table> | <b>id</b> | <b>name</b> | <b>money</b> | <b>date</b> | 7 | NULL | 5000 | 2012-10-16 | 8 | NULL | 10000 | 2012-10-16 |
| <b>id</b>  | <b>name</b> | <b>money</b> | <b>date</b>  |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |      |            |   |      |       |            |
| 1  | Alex        | 100          | 2012-10-08   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |      |            |   |      |       |            |
| 2  | Alex        | 200          | 2012-10-08   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |      |            |   |      |       |            |
| 3  | Alex        | 1000         | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |      |            |   |      |       |            |
| 4  | Alex        | 2000         | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |      |            |   |      |       |            |
| 5  | John        | 10000        | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |      |            |   |      |       |            |
| 6  | John        | 20000        | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |      |            |   |      |       |            |
| 7  | NULL        | 5000         | 2012-10-16   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |      |            |   |      |       |            |
| 8  | NULL        | 10000        | 2012-10-16   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |      |            |   |      |       |            |
| <b>id</b>  | <b>name</b> | <b>money</b> | <b>date</b>  |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |      |            |   |      |       |            |
| 7  | NULL        | 5000         | 2012-10-16   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |      |            |   |      |       |            |
| 8  | NULL        | 10000        | 2012-10-16   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |  |           |             |              |             |   |      |      |            |   |      |       |            |

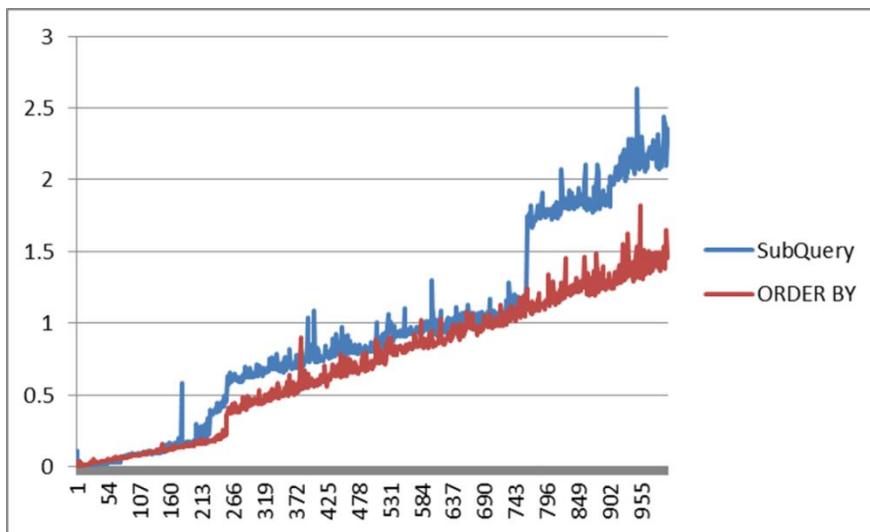
Q: Что быстрее работает?

```
SELECT * from `t_small_big` where `money` = (SELECT MAX(`money`) from `t_small_big`) LIMIT 1
```

или

```
SELECT * from `t_small_big` ORDER BY `money` DESC LIMIT 1
```

A: Проверим! Да-да, у нас есть всё та же отличная таблица, на которой мы выполняем наши тесты. С шагом в тысячу записей дойдём до миллиона, выполняя эти запросы (см. текст запросов в вопросе).



Итак, вариант с ORDER BY явно быстрее. Также следует помнить, что на скорость выполнения подобных запросов могут сильно повлиять индексы.

### 6.1.16. SELECT: [PROCEDURE procedure\_name(argument\_list)]

```
SELECT  
[ALL | DISTINCT | DISTINCTROW ]  
[HIGH_PRIORITY]  
[STRAIGHT_JOIN]  
[SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]  
[SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]  
select_expr [, select_expr ...]  
[PREVIOUS | RECURSIVE]  
[WHERE where_condition]  
[GROUP BY (col_name | expr | position)  
[ASC | DESC], ... [WITH ROLLUP]]  
[HAVING having_condition]  
[ORDER BY (col_name | expr | position)  
[ASC | DESC], ...]  
[LIMIT (offset[, row_count] | row_count OFFSET offset)]  
[PROCEDURE procedure_name(argument_list)]  
[INTO OUTFILE 'file_name'  
[COLS | LINES] SET charset_name]  
export_option  
| INTO DUMPFILE 'file_name'  
| INTO var_name [, var_name]]  
[FOR UPDATE | LOCK IN SHARE MODE]
```

Указание процедуры, которая обрабатывает результаты запроса.

Единственным примером, указанным в руководстве по MySQL, является использование ANALYSE:

<http://dev.mysql.com/doc/refman/5.5/en/procedure-analyse.html>

Все попытки «скормить» MySQL другие хранимые процедуры, которые прекрасно работают через CALL, в этом контексте приводят к сообщению типа:

ERROR 1106 (42000): Unknown procedure 'Proc';

Здесь

<http://stackoverflow.com/questions/8290157/calling-a-user-defined-storedprocedure-from-select-statement-mysql>

утверждают, что так в MySQL можно обращаться только к системным процедурам.

### 6.1.17. SELECT: [INTO OUTFILE ...]

```

SELECT
[ALL | DISTINCT | DISTINCTROW ]
[HIGH_PRIORITY]
[DYNAMIC_RESULTSET]
[SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
[SQL_CACHE] [SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
[sql_table, select_expr ...]
[FROM table_references]
[WHERE where_condition]
[ASC | DESC | [WITH] COLUMNS]
[HAVING where_condition]
[ORDER BY (col_name | expr | position)
[ASC | DESC] [WITH TIES]]
[LIMIT offset[, row_count | row_count OFFSET offset]]
[PROCEDURE procedure_name(argument_list)]
[INTO OUTFILE 'file_name'
[CHARACTER SET charset_name]
export_options
[ INTO DUMPFILE 'file_name'
| INTO var_name [, var_name]]
[FOR UPDATE | LOCK IN SHARE MODE]

```

Экспорт результатов выборки в выходной файл. Быстрый способ «выгрузить» результаты выборки для дальнейшего анализа.

Быстрый способ выгрузить таблицу в CSV-файл:

| Исходная таблица   | Запрос      | Результат    |              |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |   |
|--|-------------|--------------|--------------|-------------|---|------|-----|------------|---|------|-----|------------|---|------|------|------------|---|------|------|------------|---|------|-------|------------|---|------|-------|------------|---|------|------|------------|---|------|-------|------------|--|---|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>money</b></th><th><b>date</b></th></tr> </thead> <tbody> <tr><td>1</td><td>Alex</td><td>100</td><td>2012-10-08</td></tr> <tr><td>2</td><td>Alex</td><td>200</td><td>2012-10-08</td></tr> <tr><td>3</td><td>Alex</td><td>1000</td><td>2012-10-07</td></tr> <tr><td>4</td><td>Alex</td><td>2000</td><td>2012-10-07</td></tr> <tr><td>5</td><td>John</td><td>10000</td><td>2012-10-07</td></tr> <tr><td>6</td><td>John</td><td>20000</td><td>2012-10-07</td></tr> <tr><td>7</td><td>NULL</td><td>5000</td><td>2012-10-16</td></tr> <tr><td>8</td><td>NULL</td><td>10000</td><td>2012-10-16</td></tr> </tbody> </table> | <b>id</b>   | <b>name</b>  | <b>money</b> | <b>date</b> | 1 | Alex | 100 | 2012-10-08 | 2 | Alex | 200 | 2012-10-08 | 3 | Alex | 1000 | 2012-10-07 | 4 | Alex | 2000 | 2012-10-07 | 5 | John | 10000 | 2012-10-07 | 6 | John | 20000 | 2012-10-07 | 7 | NULL | 5000 | 2012-10-16 | 8 | NULL | 10000 | 2012-10-16 | <pre> SELECT * FROM `payment_with_date` INTO OUTFILE 'c:/file2.csv' CHARACTER SET 'latin1' FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '' </pre> | 1,"Alex",100,"2012-10-08"<br>2,"Alex",200,"2012-10-08"<br>3,"Alex",1000,"2012-10-07"<br>4,"Alex",2000,"2012-10-07"<br>5,"John",10000,"2012-10-07"<br>6,"John",20000,"2012-10-07"<br>7,\N,5000,"2012-10-16"<br>8,\N,10000,"2012-10-16" |
| <b>id</b>  | <b>name</b> | <b>money</b> | <b>date</b>  |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |   |
| 1  | Alex        | 100          | 2012-10-08   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |   |
| 2  | Alex        | 200          | 2012-10-08   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |   |
| 3  | Alex        | 1000         | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |   |
| 4  | Alex        | 2000         | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |   |
| 5  | John        | 10000        | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |   |
| 6  | John        | 20000        | 2012-10-07   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |   |
| 7  | NULL        | 5000         | 2012-10-16   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |   |
| 8  | NULL        | 10000        | 2012-10-16   |             |   |      |     |            |   |      |     |            |   |      |      |            |   |      |      |            |   |      |       |            |   |      |       |            |   |      |      |            |   |      |       |            |  |   |

### 6.1.18. SELECT: [INTO DUMPFILE ...]

```

SELECT
[ALL | DISTINCT | DISTINCTROW ]
[HIGH_PRIORITY]
[STRAIGHT_JOIN]
[SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
[SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
[select_expr [, select_expr ...]]
[PREVIOUS | REVERSE | REVERSE_SORT]
[WHERE where_condition]
[GROUP BY (col_name | expr | position)
[ASC | DESC[, ...]] WITH ROLLUP]
[HAVING having_condition]
[ORDER BY (col_name | expr | position)
[ASC | DESC[, ...]]
[LIMIT limit_count | row_count OFFSET offset]]
[PROCEDURE procedure_name(argument_list)]
[NOOUTFILE | file_name]
[CONNECTION connection_name]
[CHARACTER SET charset_name]
[export_options]

```

| **INTO DUMPFILE 'file\_name'**

[INTO var\_name [, var\_name]]

[FOR UPDATE | LOCK IN SHARE MODE]

Экспорт одного значения («одной ячейки») выборки в выходной файл. Быстрый способ «выгрузить» из БД бинарные данные.

Пример выгрузки картинки, бинарного документа и т.п.

| Исходная таблица   | Запрос           | Результат |   |                  |   |   |
|--|------------------|-----------|---|------------------|---|---|
| <table border="1"> <thead> <tr> <th>doc_id</th><th>doc_data</th></tr> </thead> <tbody> <tr> <td>1</td><td>[BLOB - 48.5KiB]</td></tr> </tbody> </table> | doc_id           | doc_data  | 1 | [BLOB - 48.5KiB] | <pre>SELECT `doc_data` FROM `storage` WHERE `doc_id`=1 INTO DUMPFILE 'c:/pic.jpg'</pre> |  |
| doc_id   | doc_data         |           |   |                  |   |   |
| 1  | [BLOB - 48.5KiB] |           |   |                  |   |   |

### 6.1.19. SELECT: [INTO var\_name [, var\_name]]]

```

SELECT
[ALL | DISTINCT | DISTINCTROW ]
[HIGH_PRIORITY]
[STRAIGHT_JOIN]
[SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
[SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
[select_expr [, select_expr ...]]
[PREVIOUS | RECURSIVE]
[WHERE where_condition]
[GROUP BY col_name | expr | position]
[ASC | DESC[, ...] [WITH ROLLUP]]
[HAVING having_condition]
[ORDER BY col_name | expr | position]
[ASC | DESC[, ...]
[LIMIT row_count | row_count OFFSET offset]
[PROCEDURE procedure_name(argument_list)]
[INTO OUTFILE 'file_name'
[CHARACTER SET charset_name]
[export_options]
| INTO DUMPFILE 'file_name'
[FOR UPDATE | LOCK IN SHARE MODE]
| INTO var_name [, var_name]]

```

Экспорт значений одного или нескольких полей в переменные для использования в последующих запросах. Запрос, результаты которого экспортируются в переменные, должен возвращать ОДИН ряд.

Иногда удобно предварительно подготовить данные в переменных, а затем использовать эти переменные в запросе:

| Исходная таблица   | Запрос       | Результат   |              |              |   |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |      |            |     |   |     |            |     |   |     |            |     |   |     |            |     |   |        |            |      |   |   |             |              |        |      |      |    |
|--|--------------|-------------|--------------|--------------|---|------|------------|----|---|------|------------|----|---|------|------------|----|---|------|------------|-----|---|------|------------|-----|---|-----|------------|-----|---|-----|------------|-----|---|-----|------------|-----|---|--------|------------|------|---|---|-------------|--------------|--------|------|------|----|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>date</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr><td>1</td><td>Alex</td><td>2012-08-21</td><td>70</td></tr> <tr><td>2</td><td>Jack</td><td>2012-07-18</td><td>60</td></tr> <tr><td>3</td><td>John</td><td>2012-07-02</td><td>50</td></tr> <tr><td>4</td><td>Alex</td><td>2012-05-17</td><td>200</td></tr> <tr><td>5</td><td>Alex</td><td>2012-05-07</td><td>500</td></tr> <tr><td>6</td><td>Joe</td><td>2012-04-27</td><td>150</td></tr> <tr><td>7</td><td>Joe</td><td>2012-04-20</td><td>150</td></tr> <tr><td>8</td><td>Joe</td><td>2012-04-10</td><td>150</td></tr> <tr><td>9</td><td>Daniel</td><td>2012-04-09</td><td>1500</td></tr> </tbody> </table> | <b>id</b>    | <b>name</b> | <b>date</b>  | <b>money</b> | 1 | Alex | 2012-08-21 | 70 | 2 | Jack | 2012-07-18 | 60 | 3 | John | 2012-07-02 | 50 | 4 | Alex | 2012-05-17 | 200 | 5 | Alex | 2012-05-07 | 500 | 6 | Joe | 2012-04-27 | 150 | 7 | Joe | 2012-04-20 | 150 | 8 | Joe | 2012-04-10 | 150 | 9 | Daniel | 2012-04-09 | 1500 | <pre> SELECT MAX(`money`), MIN(`money`) FROM `payment_archive` INTO @max, @min;  SELECT `name`, `money` from `payment_archive` where `money` = @max OR `money` = @min ORDER BY `money` DESC; </pre> | <table border="1"> <thead> <tr> <th><b>name</b></th><th><b>money</b></th></tr> </thead> <tbody> <tr><td>Daniel</td><td>1500</td></tr> <tr><td>John</td><td>50</td></tr> </tbody> </table> | <b>name</b> | <b>money</b> | Daniel | 1500 | John | 50 |
| <b>id</b>  | <b>name</b>  | <b>date</b> | <b>money</b> |              |   |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |      |            |     |   |     |            |     |   |     |            |     |   |     |            |     |   |        |            |      |   |   |             |              |        |      |      |    |
| 1  | Alex         | 2012-08-21  | 70           |              |   |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |      |            |     |   |     |            |     |   |     |            |     |   |     |            |     |   |        |            |      |   |   |             |              |        |      |      |    |
| 2  | Jack         | 2012-07-18  | 60           |              |   |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |      |            |     |   |     |            |     |   |     |            |     |   |     |            |     |   |        |            |      |   |   |             |              |        |      |      |    |
| 3  | John         | 2012-07-02  | 50           |              |   |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |      |            |     |   |     |            |     |   |     |            |     |   |     |            |     |   |        |            |      |   |   |             |              |        |      |      |    |
| 4  | Alex         | 2012-05-17  | 200          |              |   |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |      |            |     |   |     |            |     |   |     |            |     |   |     |            |     |   |        |            |      |   |   |             |              |        |      |      |    |
| 5  | Alex         | 2012-05-07  | 500          |              |   |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |      |            |     |   |     |            |     |   |     |            |     |   |     |            |     |   |        |            |      |   |   |             |              |        |      |      |    |
| 6  | Joe          | 2012-04-27  | 150          |              |   |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |      |            |     |   |     |            |     |   |     |            |     |   |     |            |     |   |        |            |      |   |   |             |              |        |      |      |    |
| 7  | Joe          | 2012-04-20  | 150          |              |   |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |      |            |     |   |     |            |     |   |     |            |     |   |     |            |     |   |        |            |      |   |   |             |              |        |      |      |    |
| 8  | Joe          | 2012-04-10  | 150          |              |   |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |      |            |     |   |     |            |     |   |     |            |     |   |     |            |     |   |        |            |      |   |   |             |              |        |      |      |    |
| 9  | Daniel       | 2012-04-09  | 1500         |              |   |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |      |            |     |   |     |            |     |   |     |            |     |   |     |            |     |   |        |            |      |   |   |             |              |        |      |      |    |
| <b>name</b>  | <b>money</b> |             |              |              |   |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |      |            |     |   |     |            |     |   |     |            |     |   |     |            |     |   |        |            |      |   |   |             |              |        |      |      |    |
| Daniel   | 1500         |             |              |              |   |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |      |            |     |   |     |            |     |   |     |            |     |   |     |            |     |   |        |            |      |   |   |             |              |        |      |      |    |
| John   | 50           |             |              |              |   |      |            |    |   |      |            |    |   |      |            |    |   |      |            |     |   |      |            |     |   |     |            |     |   |     |            |     |   |     |            |     |   |        |            |      |   |   |             |              |        |      |      |    |

### 6.1.20. SELECT: [FOR UPDATE | LOCK IN SHARE MODE]]

```

SELECT
[ALL | DISTINCT | DISTINCTROW ]
[HIGH_PRIORITY]
[STRAIGHT_JOIN]
[SQL_SMALL_RESULT] [SQL_BIO_RESULT] [SQL_BUFFER_RESULT]
[SQL_CACHE] [SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
[select_expr [, select_expr ...]]
[PREVIOUS | REINCTER]
[WHERE where_condition]
[GROUP BY (col_name [, expr | position])
[AS [S]C[, ...] (WITH|NOCOLLUP)]]
[HAVING having_condition]
[ORDER BY (col_name [, expr | position])
[ASC | DESC[, ...]]
[LIMIT row_count | row_count OFFSET offset]]
[PROCEDURE procedure_name(argument_list)]
[INTO OUTFILE 'file_name'
[CHARACTER SET charset_name]
[OPTIONAL]
export_options
| INTO DUMPFILE 'file_name'
| INTO var_name [, var_name]]

```

[FOR UPDATE | LOCK IN SHARE MODE]]

Данные команды позволяют управлять степенью изолированности выполняемой транзакции от вмешательства других транзакций.

Подробности: <http://dev.mysql.com/doc/refman/5.5/en/innodb-locking-reads.html>



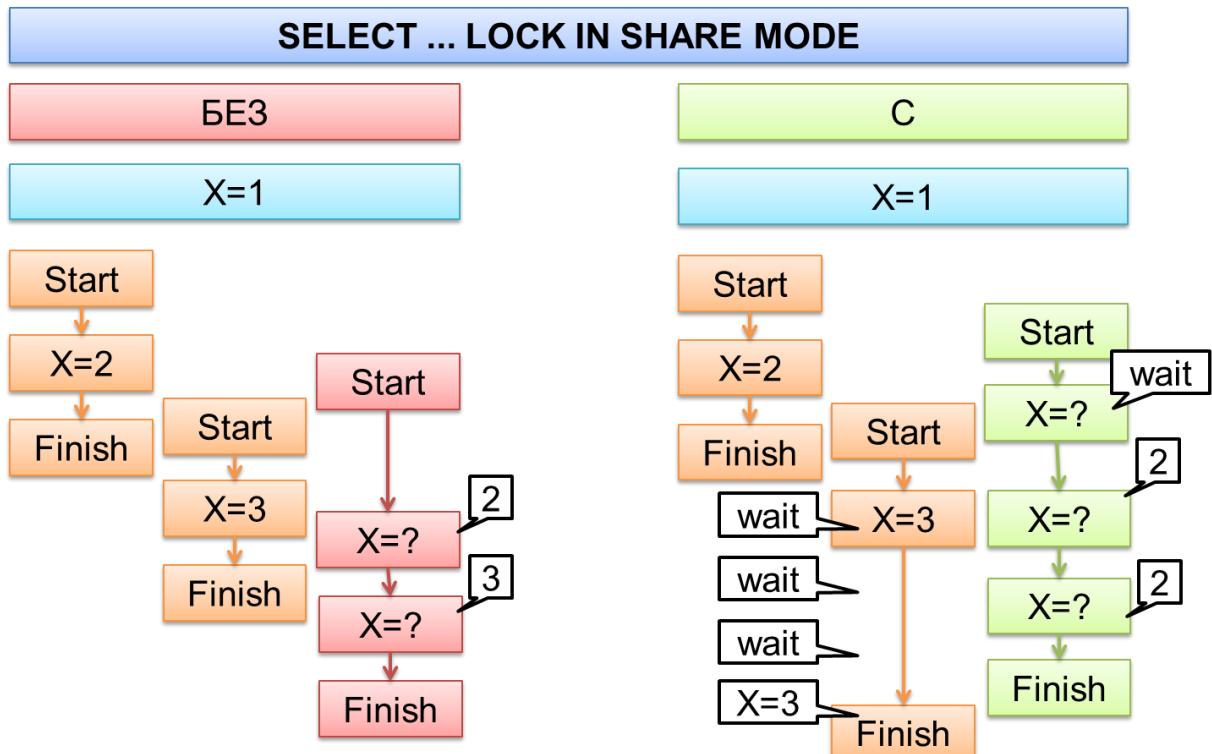
**В руководстве сказано...**

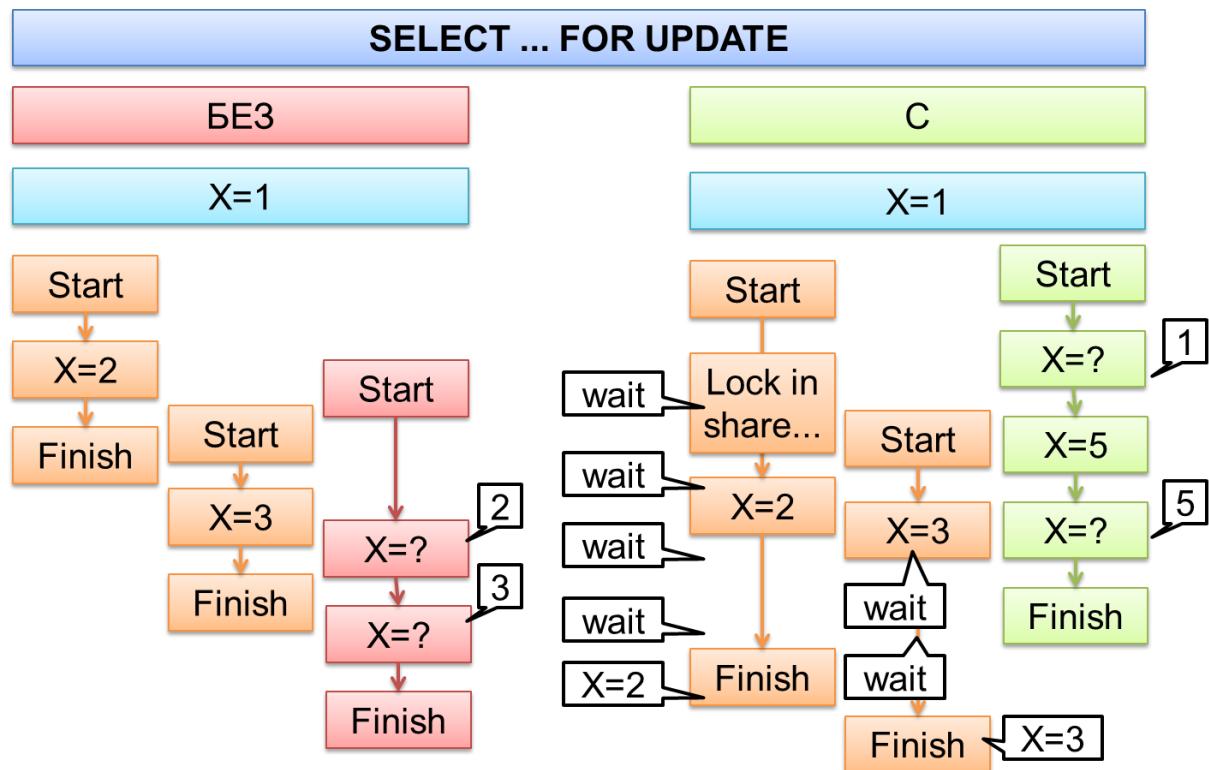
При **SELECT ... LOCK IN SHARE MODE** другие транзакции могут читать обрабатываемые вашей транзакцией ряды, но не могут их модифицировать. Если в момент старта вашей транзакции какие-то нужные ей ряды модифицировались, она ожидает завершения модифицирующей транзакции, а затем использует новые значения.

Применение **SELECT ... FOR UPDATE** блокирует ряды так, что другие транзакции не могут их модифицировать, выставляя на них **LOCK IN SHARE MODE** или даже читать их (зависит от уровня изолированности транзакций). **Этот режим доступен только в случае, если автокоммит транзакций отключён через START TRANSACTION или SET autocommit=0.**

Рекомендуется использовать при обработке любых иерархических данных (деревья, графы и т.д.)

Поясним логику работы этих конструкций на картинках.





На этом мы заканчиваем рассмотрение общего синтаксиса оператора SELECT и переходим к частным случаям 😊.

## 6.2. «Подсказки» по индексам

### 6.2.1. Обзор структуры и краткие пояснения

Оптимизатор запросов всегда пытается найти лучший (самый быстрый и не затратный по расходу памяти) способ выполнения запроса.

В сложных случаях оптимизатор может ошибаться и выбирать плохой (не устраивающий нас) вариант.

Мы можем подсказать оптимизатору, какие индексы использовать, а какие – нет.

Итак, вся структура «подсказок» по индексам выглядит следующим образом:

```
tbl_name [[AS] alias] [index_hint_list]

index_hint_list:
    index_hint [, index_hint] ...

index_hint:
    USE {INDEX|KEY}
        [FOR {JOIN|ORDER BY|GROUP BY}] ([index_list])
    | IGNORE {INDEX|KEY}
        [FOR {JOIN|ORDER BY|GROUP BY}] (index_list)
    | FORCE {INDEX|KEY}
        [FOR {JOIN|ORDER BY|GROUP BY}] (index_list)

index_list:
    index_name [, index_name] ...
```

### 6.2.2. index\_hint: USE | IGNORE | FORCE

```
tbl_name [[AS] alias] [index_hint_list]
index_hint_list:
    index_hint [, index_hint] ...
index_hint:
    USE {INDEX|KEY}
        [FOR {JOIN|ORDER BY|GROUP BY}] ([index_list])
    | IGNORE {INDEX|KEY}
        [FOR {JOIN|ORDER BY|GROUP BY}] (index_list)
    | FORCE {INDEX|KEY}
        [FOR {JOIN|ORDER BY|GROUP BY}] (index_list)
index_list:
    index_name [, index_name] ...
```

Поскольку общий синтаксис работы с таблицами мы рассмотрели в разделе, посвящённом SELECT, здесь мы сконцентрируемся на **ключевых словах USE, IGNORE, FORCE** и указании области действия «подсказки» (FOR ...).

|        |   |
|--------|---|
| USE    | Использовать <b>только перечисленные</b> индексы. Можно указать пустой список, что эквивалентно IGNORE с перечислением всех индексов.   |
| IGNORE | НЕ использовать перечисленные индексы.  |
| FORCE  | Указание того факта, что полный просмотр таблицы разрешается использовать ТОЛЬКО в случае, если ни один из указанных индексов не может быть использован для поиска требуемых записей. |

**Q:** Как узнать, какие у таблицы есть индексы?

**A:** SHOW INDEX FROM `tablename`

```
SHOW INDEX FROM `computers`
```

| Table     | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null  | Index_type | Comment | Index_comment |
|-----------|------------|----------|--------------|-------------|-----------|-------------|----------|--------|-------|------------|---------|---------------|
| computers | 0          | PRIMARY  | 1            | c_id        | A         | 5           | NULL     | NULL   | BTREE |            |         |               |
| computers | 1          | c_room   | 1            | c_room      | A         | 5           | NULL     | NULL   | YES   | BTREE      |         |               |

**Q:** Как узнать, какие индексы и для чего используются при выполнении запроса?

**A:** EXPLAIN ваш\_запрос

```
EXPLAIN SELECT * FROM `computers` WHERE `c_id`=1
```

| id | select_type | table     | type  | possible_keys | key     | key_len | ref   | rows | Extra |
|----|-------------|-----------|-------|---------------|---------|---------|-------|------|-------|
| 1  | SIMPLE      | computers | const | PRIMARY       | PRIMARY | 4       | const | 1    |       |

**Q:** Как в списке индексов указать первичный ключ таблицы?

**A:** PRIMARY

```
EXPLAIN SELECT * FROM `computers` IGNORE INDEX (PRIMARY)
WHERE `c_id`=1
```

| id | select_type | table     | type | possible_keys | key  | key_len | ref  | rows | Extra         |
|----|-------------|-----------|------|---------------|------|---------|------|------|---------------|
| 1  | SIMPLE      | computers | ALL  |               | NULL | NULL    | NULL | NULL | 5 Using where |

### 6.2.3. index\_hint: [FOR {JOIN|ORDER BY|GROUP BY}]

```

tbl_name [[AS] alias] [index_hint_list]
index_hint_list:
    index_hint [, index_hint] ...
index_hint:
    USE {INDEX|KEY}
    [FOR {JOIN|ORDER BY|GROUP BY}] ([index_list])
    IGNORE {INDEX|KEY}
    [FOR {JOIN|ORDER BY|GROUP BY}] (index_list)
    FORCE {INDEX|KEY}
    [FOR {JOIN|ORDER BY|GROUP BY}] (index_list)
index_list:
    index_name [, index_name] ...

```

Поскольку общий синтаксис работы с таблицами мы рассмотрели в разделе, посвящённом SELECT, здесь мы сконцентрируемся на ключевых словах USE, IGNORE, FORCE и **указании области действия «подсказки» (FOR ...)**.

|          |   |
|----------|---|
| JOIN     | «Подсказка» относится к JOIN-части запроса.     |
| ORDER BY | «Подсказка» относится к ORDER BY части запроса. |
| GROUP BY | «Подсказка» относится к GROUP BY части запроса. |

Таким образом мы можем более тонко управлять поведением оптимизатора относительно отдельных частей запроса.

Создадим таблицу:

```

CREATE TABLE `idxtbl` (
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `status` enum('0','1') NOT NULL,
    `name` varchar(100) NOT NULL,
    PRIMARY KEY (`id`)
) ENGINE=InnoDB
AUTO_INCREMENT=1
DEFAULT CHARSET=utf8

```

Добавим в неё 10+ миллионов записей и создадим индексы:

```

CREATE INDEX `idx1` ON `idxtbl` (`status`, `name`)
CREATE INDEX `idx2` ON `idxtbl` (`name`)

```

Проверим, как будут выполняться запросы:

```
EXPLAIN SELECT * FROM `idxtbl` WHERE `status` = '1'
```

| <b>id</b> | <b>select_type</b> | <b>table</b> | <b>type</b> | <b>possible_keys</b> | <b>key</b> | <b>key_len</b> | <b>ref</b> | <b>rows</b> | <b>Extra</b>             |
|-----------|--------------------|--------------|-------------|----------------------|------------|----------------|------------|-------------|--------------------------|
| 1         | SIMPLE             | idxtbl       | ref         | idx1                 | idx1       | 1              | const      | 5008859     | Using where; Using index |

```
EXPLAIN SELECT * FROM `idxtbl` IGNORE INDEX (`idx1`) WHERE `status`='1'
```

| <b>id</b> | <b>select_type</b> | <b>table</b> | <b>type</b> | <b>possible_keys</b> | <b>key</b> | <b>key_len</b> | <b>ref</b> | <b>rows</b> | <b>Extra</b> |
|-----------|--------------------|--------------|-------------|----------------------|------------|----------------|------------|-------------|--------------|
| 1         | SIMPLE             | idxtbl       | ALL         |                      | NULL       | NULL           | NULL       | NULL        | Using where  |

```
EXPLAIN SELECT * FROM `idxtbl` FORCE INDEX (`idx2`) WHERE `status`='1'
```

| <b>id</b> | <b>select_type</b> | <b>table</b> | <b>type</b> | <b>possible_keys</b> | <b>key</b> | <b>key_len</b> | <b>ref</b> | <b>rows</b> | <b>Extra</b> |
|-----------|--------------------|--------------|-------------|----------------------|------------|----------------|------------|-------------|--------------|
| 1         | SIMPLE             | idxtbl       | ALL         |                      | NULL       | NULL           | NULL       | NULL        | Using where  |

#### **6.2.4. Краткие итоги**

1. В общем случае оптимизатор достаточно хорошо «угадывает», какие индексы использовать.
  2. Если оптимизатор сам решает не использовать индексы, его можно «заставить», но далеко не факт, что это даст эффект (как правило, прирост производительности оказывается в районе долей процентов).
  3. Реальную пользу «подсказки» по индексам приносят в случае действительно сложных запросов, которые нужно многократно проверять через EXPLAIN и реальными тестами.

## 6.3. Объединение запросов (UNION)

### 6.3.1. Обзор структуры и краткие пояснения

Объединение запросов с применением UNION используется для объединения результатов нескольких запросов в один итоговый результат.

Итак, вся структура объединения запросов выглядит следующим образом:

```
SELECT ...
UNION [ALL | DISTINCT] SELECT ...
[UNION [ALL | DISTINCT] SELECT ...]
```

Дополнительные примеры использования UNION см. в разделе, посвящённом SELECT.

### 6.3.2. SELECT ... UNION

**SELECT ...  
UNION [ALL | DISTINCT] SELECT ...  
[UNION [ALL | DISTINCT] SELECT ...]**

Используется для объединения результатов нескольких запросов в один итоговый результат.

 **В руководстве сказано...**

У всех подзапросов должно быть одинаковое количество извлекаемых колонок, имена которых определяются первым подзапросом. Если типы данных, возвращаемых в колонку разными подзапросами, не совпадают, MySQL учитывает это и сохраняет все значения. По умолчанию UNION работает в режиме DISTINCT.

Множество дополнительных мелких деталей см. здесь:

<http://dev.mysql.com/doc/refman/5.5/en/union.html>

### 6.3.3. Пример использования UNION

Допустим, у нас есть две таблицы платежей – операционная и архивная.

| <b>id</b> | <b>name</b> | <b>date</b> | <b>money</b> | <b>id</b> | <b>name</b> | <b>date</b> | <b>money</b> |
|-----------|-------------|-------------|--------------|-----------|-------------|-------------|--------------|
| 1001      | John        | 2012-10-15  | 100          | 1         | Alex        | 2012-08-21  | 70           |
| 1002      | Jack        | 2012-10-16  | 200          | 2         | Jack        | 2012-07-18  | 60           |
|           |             |             |              | 3         | John        | 2012-07-02  | 50           |
|           |             |             |              | 4         | Alex        | 2012-05-17  | 200          |
|           |             |             |              | 5         | Alex        | 2012-05-07  | 500          |
|           |             |             |              | 6         | Joe         | 2012-04-27  | 150          |
|           |             |             |              | 7         | Joe         | 2012-04-20  | 150          |
|           |             |             |              | 8         | Joe         | 2012-04-10  | 150          |
|           |             |             |              | 9         | Daniel      | 2012-04-09  | 1500         |

Выяснить сумму платежей по каждому пользователю за всё время можно так:

```
SELECT `name` , SUM(`money`)
FROM (
    SELECT `name` , `money`
    FROM `payment_operational`
    UNION ALL
    SELECT `name` , `money`
    FROM `payment_archive`
) AS `X`
GROUP BY `name`
```

| <b>name</b> | <b>SUM(`money`)</b> |
|-------------|---------------------|
| Alex        | 770                 |
| Daniel      | 1500                |
| Jack        | 260                 |
| Joe         | 450                 |
| John        | 150                 |

А что получится, если выполнить вот такой запрос (сначала выборка и группировка, потом объединение)?

```
SELECT `name` , SUM(`money`)
FROM `payment_operational`
GROUP BY `name`
UNION ALL
SELECT `name` , SUM(`money`)
FROM `payment_archive`
```

| name   | SUM(`money`) |
|--------|--------------|
| Jack   | 200          |
| John   | 100          |
| Alex   | 770          |
| Daniel | 1500         |
| Jack   | 60           |
| Joe    | 450          |
| John   | 50           |



Как видно, итоговый результат содержит «сборку» двух независимых вычислений сумм, а не итоговые суммы.

#### 6.3.4. Краткие итоги

1. При всей предельной простоте UNION может преподнести неприятные сюрпризы.
2. В большинстве случаев, если вам кажется, что надо использовать UNION, вы чего-то не предусмотрели или на уровне создания модели БД, или на стадии обдумывания задачи, для решения которой пишете запрос ☺.

## 6.4. Запросы на объединение (JOIN)

### 6.4.1. Назначение и общая структура JOIN

Запросы на объединение выполняются в случае, когда данные, необходимые для получения конечной выборки, находятся в разных таблицах (или в одной таблице, представляя собой иерархическую структуру) и при этом между такими данными существует некая логическая или семантическая связь.

Общая структура JOIN такова:

```
table_references:
    table_reference [, table_reference] ...

table_reference:
    table_factor
    | join_table

table_factor:
    tbl_name [[AS] alias] [index_hint_list]
    | table_subquery [AS] alias
    | ( table_references )
    | { OJ table_reference LEFT OUTER JOIN table_reference
        ON conditional_expr }

join_table:
    table_reference [INNER | CROSS] JOIN table_factor [join_condition]
    | table_reference STRAIGHT_JOIN table_factor
    | table_reference STRAIGHT_JOIN table_factor ON conditional_expr
    | table_reference {LEFT|RIGHT} [OUTER] JOIN table_reference join_condition
    | table_reference NATURAL [{LEFT|RIGHT} [OUTER]] JOIN table_factor

join_condition:
    ON conditional_expr
    | USING (column_list)

index_hint_list:
    index_hint [, index_hint] ...

index_hint:
    USE {INDEX|KEY}
        [FOR {JOIN|ORDER BY|GROUP BY}] ([index_list])
    | IGNORE {INDEX|KEY}
        [FOR {JOIN|ORDER BY|GROUP BY}] (index_list)
    | FORCE {INDEX|KEY}
        [FOR {JOIN|ORDER BY|GROUP BY}] (index_list)

index_list:
    index_name [, index_name] ...
```

Если рассмотреть подробнее каждую секцию JOIN, получится:

```
table_references:
  table_reference [, table_reference] ...

table_reference:
  table_factor
  | join_table

table_factor:
 tbl_name [[AS] alias] [index_hint_list]
  | table_subquery [AS] alias
  | (table_references)
  | (JOIN table_reference LEFTOUTER JOIN table_reference
    ON conditional_expr)

join_table:
  table_reference INNER|CROSS|JOIN table_factor [join_condition]
  | table_reference STRAIGHT_JOIN table_factor ON conditional_expr
  | table_reference (LEFT|RIGHT|OUTER) JOIN table_reference join_condition
  | table_reference NATURAL [(LEFT|RIGHT)|OUTER] JOIN table_factor

join_condition:
  ON conditional_expr
  | USING(column_list)

index_hint:
  index_hint[, index_hint] ...

index_hint:
  USE INDEX(KEY)
  | FOR (JOIN|ORDER BY|GROUP BY) [(index_list)]
  | IGNORE (INDEXKEY)
  | FOR (JOIN|ORDER BY|GROUP BY) (index_list)
  | FORCE (INDEXKEY)
  | FOR (JOIN|ORDER BY|GROUP BY) (index_list)

index_list:
  index_name [, index_name] ...
```

```
table_references:
  table_reference [, table_reference] ...

table_reference:
  table_factor
  | join_table

table_factor:
 tbl_name [[AS] alias] [index_hint_list]
  | table_subquery [AS] alias
  | (table_references)
  | (JOIN table_reference LEFTOUTER JOIN table_reference
    ON conditional_expr)

join_table:
  table_reference INNER|CROSS|JOIN table_factor [join_condition]
  | table_reference STRAIGHT_JOIN table_factor ON conditional_expr
  | table_reference (LEFT|RIGHT|OUTER) JOIN table_reference
  | table_reference NATURAL [(LEFT|RIGHT)|OUTER] JOIN table_factor

join_condition:
  ON conditional_expr
  | USING(column_list)

index_hint:
  index_hint[, index_hint] ...

index_hint:
  USE INDEX(KEY)
  | FOR (JOIN|ORDER BY|GROUP BY) [(index_list)]
  | IGNORE (INDEXKEY)
  | FOR (JOIN|ORDER BY|GROUP BY) (index_list)
  | FORCE (INDEXKEY)
  | FOR (JOIN|ORDER BY|GROUP BY) (index_list)

index_list:
  index_name [, index_name] ...
```

```
table_references:
  table_reference [, table_reference] ...

table_reference:
  table_factor
  | join_table

table_factor:
 tbl_name [[AS] alias] [index_hint_list]
  | table_subquery [AS] alias
  | (table_references)
  | (JOIN table_reference LEFTOUTER JOIN table_reference
    ON conditional_expr)

join_table:
  table_reference INNER|CROSS|JOIN table_factor [join_condition]
  | table_reference STRAIGHT_JOIN table_factor
  | table_reference STRAIGHT_JOIN table_factor ON conditional_expr
  | table_reference (LEFT|RIGHT|OUTER) JOIN table_reference join_condition
  | table_reference NATURAL [(LEFT|RIGHT)|OUTER] JOIN table_factor

join_condition:
  ON conditional_expr
  | USING (column_list)

index_hint:
  index_hint[, index_hint] ...

index_hint:
  USE INDEX(KEY)
  | FOR (JOIN|ORDER BY|GROUP BY) [(index_list)]
  | IGNORE (INDEXKEY)
  | FOR (JOIN|ORDER BY|GROUP BY) (index_list)
  | FORCE (INDEXKEY)
  | FOR (JOIN|ORDER BY|GROUP BY) (index_list)

index_list:
  index_name [, index_name] ...
```

```
table_references:
  table_reference [, table_reference] ...

table_reference:
  table_factor
  | join_table

table_factor:
 tbl_name [[AS] alias] [index_hint_list]
  | table_subquery [AS] alias
  | (table_references)
  | (JOIN table_reference LEFTOUTER JOIN table_reference
    ON conditional_expr)

join_table:
  table_reference INNER|CROSS|JOIN table_factor [join_condition]
  | table_reference STRAIGHT_JOIN table_factor
  | table_reference STRAIGHT_JOIN table_factor ON conditional_expr
  | table_reference (LEFT|RIGHT|OUTER) JOIN table_reference join_condition
  | table_reference NATURAL [(LEFT|RIGHT)|OUTER] JOIN table_factor

join_condition:
  ON conditional_expr
  | USING(column_list)

index_hint:
  index_hint[, index_hint] ...

index_hint:
  USE INDEX(KEY)
  | FOR (JOIN|ORDER BY|GROUP BY) [(index_list)]
  | IGNORE (INDEXKEY)
  | FOR (JOIN|ORDER BY|GROUP BY) (index_list)
  | FORCE (INDEXKEY)
  | FOR (JOIN|ORDER BY|GROUP BY) (index_list)

index_list:
  index_name [, index_name] ...
```

Основная часть запроса (также известна как «JOIN-выражение»), применяется совместно с операторами SELECT, DELETE и UPDATE.

Описывает перечень таблиц и способ объединения. (Будет рассмотрено дальше на примерах.)

Описывает условие объединения. (Будет рассмотрено дальше на примерах.)

Описывает «подсказки» по использованию индексов. (См. соответствующий раздел: «Подсказки» по индексам.).

### 6.4.2. Принципы работы JOIN

JOIN позволяет выполнить запрос, объединяющий данные из одной (self-join), двух и более таблиц.

Такой запрос позволяет получить информацию на основе связей между таблицами или иных (нетривиальных) структурных особенностей модели базы данных.

В общем случае использование JOIN необходимо для денормализации с целью агрегирования или кэширования данных, распределённых по нормализованным отношениям (таблицам) базы данных.

### 6.4.3. Три простых примера использования JOIN

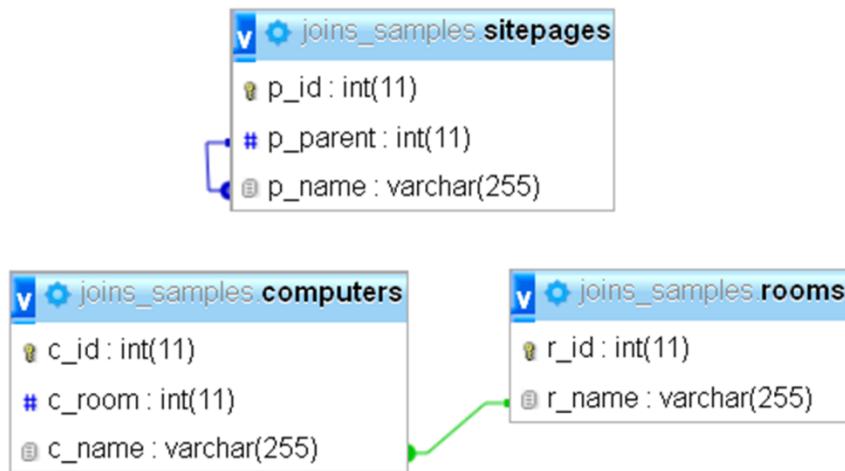
Для начала рассмотрим три простых примера использования JOIN. Для этого создадим три таблицы (`rooms`, `computers`, `sitepages`) и наполним их небольшим количеством данных.

Таблицы `rooms` и `computers` будут связаны связью «один ко многим».

В таблице `sitepages` будет рекурсивный внешний ключ (ссылка на родительскую страницу).

```
CREATE TABLE `rooms`  
(  
    `r_id` int(11) NOT NULL AUTO_INCREMENT,  
    `r_name` varchar(255) NOT NULL,  
    PRIMARY KEY (`r_id`)  
)  
ENGINE=InnoDB DEFAULT CHARSET=utf8  
  
CREATE TABLE `computers`  
(  
    `c_id` int(11) NOT NULL AUTO_INCREMENT,  
    `c_room` int(11) DEFAULT NULL,  
    `c_name` varchar(255) NOT NULL,  
    PRIMARY KEY (`c_id`),  
    KEY `c_room` (`c_room`),  
    CONSTRAINT `computers_ibfk_1`  
    FOREIGN KEY (`c_room`)  
    REFERENCES `rooms` (`r_id`)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE  
)  
ENGINE=InnoDB DEFAULT CHARSET=utf8  
  
CREATE TABLE `sitepages`  
(  
    `p_id` int(11) NOT NULL AUTO_INCREMENT,  
    `p_parent` int(11) DEFAULT NULL,  
    `p_name` varchar(255) NOT NULL,  
    PRIMARY KEY (`p_id`),  
    KEY `p_parent` (`p_parent`),  
    CONSTRAINT `sitepages_ibfk_1`  
    FOREIGN KEY (`p_parent`)  
    REFERENCES `sitepages` (`p_id`)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE  
)  
ENGINE=InnoDB DEFAULT CHARSET=utf8
```

Схема базы данных приобретает такой вид:



Таблицы с данными:

| Таблица   | Данные   |                     |          |        |                  |      |                    |   |              |                     |              |   |                     |   |      |             |   |      |             |
|---|--|---------------------|----------|--------|------------------|------|--------------------|---|--------------|---------------------|--------------|---|---------------------|---|------|-------------|---|------|-------------|
| `rooms`   | <table border="1"> <thead> <tr> <th>r_id</th> <th>r_name</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>RoomWithTwoComps</td> </tr> <tr> <td>2</td> <td>RoomWithOneComp</td> </tr> <tr> <td>3</td> <td>EmptyRoomOne</td> </tr> <tr> <td>4</td> <td>EmptyRoomTwo</td> </tr> </tbody> </table>  | r_id                | r_name   | 1      | RoomWithTwoComps | 2    | RoomWithOneComp    | 3 | EmptyRoomOne | 4                   | EmptyRoomTwo |   |                     |   |      |             |   |      |             |
| r_id  | r_name   |                     |          |        |                  |      |                    |   |              |                     |              |   |                     |   |      |             |   |      |             |
| 1   | RoomWithTwoComps   |                     |          |        |                  |      |                    |   |              |                     |              |   |                     |   |      |             |   |      |             |
| 2   | RoomWithOneComp  |                     |          |        |                  |      |                    |   |              |                     |              |   |                     |   |      |             |   |      |             |
| 3   | EmptyRoomOne   |                     |          |        |                  |      |                    |   |              |                     |              |   |                     |   |      |             |   |      |             |
| 4   | EmptyRoomTwo   |                     |          |        |                  |      |                    |   |              |                     |              |   |                     |   |      |             |   |      |             |
| `computers`   | <table border="1"> <thead> <tr> <th>c_id</th> <th>c_room</th> <th>c_name</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>FirstCompInRoomOne</td> </tr> <tr> <td>2</td> <td>1</td> <td>SecondCompInRoomOne</td> </tr> <tr> <td>3</td> <td>2</td> <td>SingleCompInRoomTwo</td> </tr> <tr> <td>4</td> <td>NULL</td> <td>FreeCompOne</td> </tr> <tr> <td>5</td> <td>NULL</td> <td>FreeCompTwo</td> </tr> </tbody> </table>  | c_id                | c_room   | c_name | 1                | 1    | FirstCompInRoomOne | 2 | 1            | SecondCompInRoomOne | 3            | 2 | SingleCompInRoomTwo | 4 | NULL | FreeCompOne | 5 | NULL | FreeCompTwo |
| c_id  | c_room   | c_name              |          |        |                  |      |                    |   |              |                     |              |   |                     |   |      |             |   |      |             |
| 1   | 1  | FirstCompInRoomOne  |          |        |                  |      |                    |   |              |                     |              |   |                     |   |      |             |   |      |             |
| 2   | 1  | SecondCompInRoomOne |          |        |                  |      |                    |   |              |                     |              |   |                     |   |      |             |   |      |             |
| 3   | 2  | SingleCompInRoomTwo |          |        |                  |      |                    |   |              |                     |              |   |                     |   |      |             |   |      |             |
| 4   | NULL   | FreeCompOne         |          |        |                  |      |                    |   |              |                     |              |   |                     |   |      |             |   |      |             |
| 5   | NULL   | FreeCompTwo         |          |        |                  |      |                    |   |              |                     |              |   |                     |   |      |             |   |      |             |
| `sitepages`<br>(Справа показана сохранённая в таблице структура.) | <table border="1"> <thead> <tr> <th>p_id</th> <th>p_parent</th> <th>p_name</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>NULL</td> <td>Main</td> </tr> <tr> <td>2</td> <td>1</td> <td>Contacts</td> </tr> <tr> <td>3</td> <td>1</td> <td>Documents</td> </tr> <tr> <td>4</td> <td>3</td> <td>Templates</td> </tr> <tr> <td>5</td> <td>3</td> <td>Examples</td> </tr> </tbody> </table> <pre> graph TD     Main[Main] --- Contacts[Contacts]     Main --- Documents[Documents]     Documents --- Templates[Templates]     Documents --- Examples[Examples]     </pre> | p_id                | p_parent | p_name | 1                | NULL | Main               | 2 | 1            | Contacts            | 3            | 1 | Documents           | 4 | 3    | Templates   | 5 | 3    | Examples    |
| p_id  | p_parent   | p_name              |          |        |                  |      |                    |   |              |                     |              |   |                     |   |      |             |   |      |             |
| 1   | NULL   | Main                |          |        |                  |      |                    |   |              |                     |              |   |                     |   |      |             |   |      |             |
| 2   | 1  | Contacts            |          |        |                  |      |                    |   |              |                     |              |   |                     |   |      |             |   |      |             |
| 3   | 1  | Documents           |          |        |                  |      |                    |   |              |                     |              |   |                     |   |      |             |   |      |             |
| 4   | 3  | Templates           |          |        |                  |      |                    |   |              |                     |              |   |                     |   |      |             |   |      |             |
| 5   | 3  | Examples            |          |        |                  |      |                    |   |              |                     |              |   |                     |   |      |             |   |      |             |

**Пример 1: показать список комнат и поставленных в них компьютеров.**

| Запрос           | <pre>SELECT `r_name`, `c_name` FROM `rooms` JOIN `computers` ON `r_id` = `c_room`</pre>  |  |        |        |                  |                    |                  |                     |                 |                     |
|------------------|--|--|--------|--------|------------------|--------------------|------------------|---------------------|-----------------|---------------------|
| Результат        | <table border="1"> <thead> <tr> <th>r_name</th> <th>c_name</th> </tr> </thead> <tbody> <tr> <td>RoomWithTwoComps</td> <td>FirstCompInRoomOne</td> </tr> <tr> <td>RoomWithTwoComps</td> <td>SecondCompInRoomOne</td> </tr> <tr> <td>RoomWithOneComp</td> <td>SingleCompInRoomTwo</td> </tr> </tbody> </table> |  | r_name | c_name | RoomWithTwoComps | FirstCompInRoomOne | RoomWithTwoComps | SecondCompInRoomOne | RoomWithOneComp | SingleCompInRoomTwo |
| r_name           | c_name   |  |        |        |                  |                    |                  |                     |                 |                     |
| RoomWithTwoComps | FirstCompInRoomOne   |  |        |        |                  |                    |                  |                     |                 |                     |
| RoomWithTwoComps | SecondCompInRoomOne  |  |        |        |                  |                    |                  |                     |                 |                     |
| RoomWithOneComp  | SingleCompInRoomTwo  |  |        |        |                  |                    |                  |                     |                 |                     |

**Пример 2: показать список ВСЕХ комнат (даже пустых) и ВСЕХ компьютеров (даже тех, что никуда не поставлены).**

| Запрос           | <pre>(SELECT `r_name`, `c_name` FROM `rooms` LEFT OUTER JOIN `computers` ON `r_id` = `c_room`) UNION DISTINCT (SELECT `r_name`, `c_name` FROM `rooms` RIGHT OUTER JOIN `computers` ON `r_id` = `c_room`)</pre>   |  |        |        |                  |                    |                  |                     |                 |                     |              |      |              |      |      |             |      |             |
|------------------|--|--|--------|--------|------------------|--------------------|------------------|---------------------|-----------------|---------------------|--------------|------|--------------|------|------|-------------|------|-------------|
| Результат        | <table border="1"> <thead> <tr> <th>r_name</th> <th>c_name</th> </tr> </thead> <tbody> <tr> <td>RoomWithTwoComps</td> <td>FirstCompInRoomOne</td> </tr> <tr> <td>RoomWithTwoComps</td> <td>SecondCompInRoomOne</td> </tr> <tr> <td>RoomWithOneComp</td> <td>SingleCompInRoomTwo</td> </tr> <tr> <td>EmptyRoomOne</td> <td>NULL</td> </tr> <tr> <td>EmptyRoomTwo</td> <td>NULL</td> </tr> <tr> <td>NULL</td> <td>FreeCompOne</td> </tr> <tr> <td>NULL</td> <td>FreeCompTwo</td> </tr> </tbody> </table> |  | r_name | c_name | RoomWithTwoComps | FirstCompInRoomOne | RoomWithTwoComps | SecondCompInRoomOne | RoomWithOneComp | SingleCompInRoomTwo | EmptyRoomOne | NULL | EmptyRoomTwo | NULL | NULL | FreeCompOne | NULL | FreeCompTwo |
| r_name           | c_name   |  |        |        |                  |                    |                  |                     |                 |                     |              |      |              |      |      |             |      |             |
| RoomWithTwoComps | FirstCompInRoomOne   |  |        |        |                  |                    |                  |                     |                 |                     |              |      |              |      |      |             |      |             |
| RoomWithTwoComps | SecondCompInRoomOne  |  |        |        |                  |                    |                  |                     |                 |                     |              |      |              |      |      |             |      |             |
| RoomWithOneComp  | SingleCompInRoomTwo  |  |        |        |                  |                    |                  |                     |                 |                     |              |      |              |      |      |             |      |             |
| EmptyRoomOne     | NULL   |  |        |        |                  |                    |                  |                     |                 |                     |              |      |              |      |      |             |      |             |
| EmptyRoomTwo     | NULL   |  |        |        |                  |                    |                  |                     |                 |                     |              |      |              |      |      |             |      |             |
| NULL             | FreeCompOne  |  |        |        |                  |                    |                  |                     |                 |                     |              |      |              |      |      |             |      |             |
| NULL             | FreeCompTwo  |  |        |        |                  |                    |                  |                     |                 |                     |              |      |              |      |      |             |      |             |

### Пример 3: показать все страницы сайта первых трёх уровней с их родителями.

| Запрос    | SELECT `level1`.`p_parent`, `level1`.`p_name`, `level1`.`p_id`, `level2`.`p_parent`, `level2`.`p_name`, `level2`.`p_id`, `level3`.`p_parent`, `level3`.`p_name` FROM `sitepages` AS `level1` LEFT OUTER JOIN `sitepages` AS `level2` ON `level1`.`p_id` = `level2`.`p_parent` LEFT OUTER JOIN `sitepages` AS `level3` ON `level2`.`p_id` = `level3`.`p_parent` WHERE ISNULL(`level1`.`p_parent`)  |          |           |          |           |          |        |      |      |   |          |      |      |      |      |   |           |   |           |      |      |   |           |   |          |
|-----------|---|----------|-----------|----------|-----------|----------|--------|------|------|---|----------|------|------|------|------|---|-----------|---|-----------|------|------|---|-----------|---|----------|
| Результат | <table border="1"> <thead> <tr> <th>p_parent</th><th>p_name</th><th>p_parent</th><th>p_name</th><th>p_parent</th><th>p_name</th></tr> </thead> <tbody> <tr> <td>NULL</td><td>Main</td><td>1</td><td>Contacts</td><td>NULL</td><td>NULL</td></tr> <tr> <td>NULL</td><td>Main</td><td>1</td><td>Documents</td><td>3</td><td>Templates</td></tr> <tr> <td>NULL</td><td>Main</td><td>1</td><td>Documents</td><td>3</td><td>Examples</td></tr> </tbody> </table> | p_parent | p_name    | p_parent | p_name    | p_parent | p_name | NULL | Main | 1 | Contacts | NULL | NULL | NULL | Main | 1 | Documents | 3 | Templates | NULL | Main | 1 | Documents | 3 | Examples |
| p_parent  | p_name  | p_parent | p_name    | p_parent | p_name    |          |        |      |      |   |          |      |      |      |      |   |           |   |           |      |      |   |           |   |          |
| NULL      | Main  | 1        | Contacts  | NULL     | NULL      |          |        |      |      |   |          |      |      |      |      |   |           |   |           |      |      |   |           |   |          |
| NULL      | Main  | 1        | Documents | 3        | Templates |          |        |      |      |   |          |      |      |      |      |   |           |   |           |      |      |   |           |   |          |
| NULL      | Main  | 1        | Documents | 3        | Examples  |          |        |      |      |   |          |      |      |      |      |   |           |   |           |      |      |   |           |   |          |

Кто-то может подумать, что это – не такие уж и простые примеры. Так вот – нет, вполне простые. Чтобы лучше понять, как и что в них работает – поэкспериментируйте. Вот вам дамп базы.

```

SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

-- 
-- Table structure for table `computers`
-- 

CREATE TABLE IF NOT EXISTS `computers` (
  `c_id` int(11) NOT NULL AUTO_INCREMENT,
  `c_room` int(11) DEFAULT NULL,
  `c_name` varchar(255) NOT NULL,
  PRIMARY KEY (`c_id`),
  KEY `c_room` (`c_room`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=6 ;

-- 
-- Dumping data for table `computers`
-- 

INSERT INTO `computers` (`c_id`, `c_room`, `c_name`) VALUES
(1, 1, 'FirstCompInRoomOne'),
(2, 1, 'SecondCompInRoomOne'),
(3, 2, 'SingleCompInRoomTwo'),
(4, NULL, 'FreeCompOne'),
(5, NULL, 'FreeCompTwo');

-- 
-- Table structure for table `rooms`
-- 

CREATE TABLE IF NOT EXISTS `rooms` (
  `r_id` int(11) NOT NULL AUTO_INCREMENT,
  `r_name` varchar(255) NOT NULL,
  PRIMARY KEY (`r_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=5 ;

-- 
-- Dumping data for table `rooms`
-- 

INSERT INTO `rooms` (`r_id`, `r_name`) VALUES
(1, 'RoomWithTwoComps'),
(2, 'RoomWithOneComp'),
(3, 'EmptyRoomOne'),
(4, 'EmptyRoomTwo');

```

```
--  
-- Table structure for table `sitepages`  
--  
  
CREATE TABLE IF NOT EXISTS `sitepages` (  
  `p_id` int(11) NOT NULL AUTO_INCREMENT,  
  `p_parent` int(11) DEFAULT NULL,  
  `p_name` varchar(255) NOT NULL,  
  PRIMARY KEY (`p_id`),  
  KEY `p_parent` (`p_parent`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=6 ;  
  
--  
-- Dumping data for table `sitepages`  
--  
  
INSERT INTO `sitepages` (`p_id`, `p_parent`, `p_name`) VALUES  
(1, NULL, 'Main'),  
(2, 1, 'Contacts'),  
(3, 1, 'Documents'),  
(4, 3, 'Templates'),  
(5, 3, 'Examples');  
  
--  
-- Constraints for table `computers`  
--  
ALTER TABLE `computers`  
  ADD CONSTRAINT `computers_ibfk_1` FOREIGN KEY (`c_room`) REFERENCES `rooms` (`r_id`) ON DELETE  
CASCADE ON UPDATE CASCADE;  
  
--  
-- Constraints for table `sitepages`  
--  
ALTER TABLE `sitepages`  
  ADD CONSTRAINT `sitepages_ibfk_1` FOREIGN KEY (`p_parent`) REFERENCES `sitepages` (`p_id`) ON  
DELETE CASCADE ON UPDATE CASCADE;  
  
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;  
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;  
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

#### 6.4.4. Основные разновидности JOIN

В только что рассмотренных примерах мы использовали несколько разновидностей JOIN'ов.

Сейчас мы рассмотрим их подробнее.

В первую очередь нас буду интересовать:

- JOIN
- LEFT JOIN
- RIGHT JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN

##### «Просто» JOIN

JOIN без дополнительных «параметров» возвращает совпадения по всем указанным столбцам (и таблицам). Если для какой-то строки в какой-то из таблиц нет совпадений, она не попадает в итоговый результат. Так, например, здесь не показаны комнаты без компьютеров и компьютеры, не установленные ни в одну из комнат.

| Запрос           | <pre>SELECT `r_name`, `c_name` FROM `rooms` JOIN `computers` ON `r_id` = `c_room`</pre>  |  |        |        |                  |                    |                  |                     |                 |                     |
|------------------|--|--|--------|--------|------------------|--------------------|------------------|---------------------|-----------------|---------------------|
| Результат        | <table border="1"> <thead> <tr> <th>r_name</th> <th>c_name</th> </tr> </thead> <tbody> <tr> <td>RoomWithTwoComps</td> <td>FirstCompInRoomOne</td> </tr> <tr> <td>RoomWithTwoComps</td> <td>SecondCompInRoomOne</td> </tr> <tr> <td>RoomWithOneComp</td> <td>SingleCompInRoomTwo</td> </tr> </tbody> </table> |  | r_name | c_name | RoomWithTwoComps | FirstCompInRoomOne | RoomWithTwoComps | SecondCompInRoomOne | RoomWithOneComp | SingleCompInRoomTwo |
| r_name           | c_name   |  |        |        |                  |                    |                  |                     |                 |                     |
| RoomWithTwoComps | FirstCompInRoomOne   |  |        |        |                  |                    |                  |                     |                 |                     |
| RoomWithTwoComps | SecondCompInRoomOne  |  |        |        |                  |                    |                  |                     |                 |                     |
| RoomWithOneComp  | SingleCompInRoomTwo  |  |        |        |                  |                    |                  |                     |                 |                     |

##### LEFT [OUTER] JOIN

LEFT JOIN возвратит строки из левой таблицы даже тогда, когда для них нет совпадения в правой таблице. Так, например, здесь показаны комнаты без компьютеров, но не показаны компьютеры, не установленные ни в одну из комнат.

| Запросы          | <pre>SELECT `r_name`, `c_name` FROM `rooms` LEFT JOIN `computers` ON `r_id` = `c_room`</pre><br><pre>SELECT `r_name`, `c_name` FROM `rooms` LEFT OUTER JOIN `computers` ON `r_id` = `c_room`</pre>   |  |        |        |                  |                    |                  |                     |                 |                     |              |      |              |      |
|------------------|--|--|--------|--------|------------------|--------------------|------------------|---------------------|-----------------|---------------------|--------------|------|--------------|------|
| Результат        | <table border="1"> <thead> <tr> <th>r_name</th> <th>c_name</th> </tr> </thead> <tbody> <tr> <td>RoomWithTwoComps</td> <td>FirstCompInRoomOne</td> </tr> <tr> <td>RoomWithTwoComps</td> <td>SecondCompInRoomOne</td> </tr> <tr> <td>RoomWithOneComp</td> <td>SingleCompInRoomTwo</td> </tr> <tr> <td>EmptyRoomOne</td> <td>NULL</td> </tr> <tr> <td>EmptyRoomTwo</td> <td>NULL</td> </tr> </tbody> </table> |  | r_name | c_name | RoomWithTwoComps | FirstCompInRoomOne | RoomWithTwoComps | SecondCompInRoomOne | RoomWithOneComp | SingleCompInRoomTwo | EmptyRoomOne | NULL | EmptyRoomTwo | NULL |
| r_name           | c_name   |  |        |        |                  |                    |                  |                     |                 |                     |              |      |              |      |
| RoomWithTwoComps | FirstCompInRoomOne   |  |        |        |                  |                    |                  |                     |                 |                     |              |      |              |      |
| RoomWithTwoComps | SecondCompInRoomOne  |  |        |        |                  |                    |                  |                     |                 |                     |              |      |              |      |
| RoomWithOneComp  | SingleCompInRoomTwo  |  |        |        |                  |                    |                  |                     |                 |                     |              |      |              |      |
| EmptyRoomOne     | NULL   |  |        |        |                  |                    |                  |                     |                 |                     |              |      |              |      |
| EmptyRoomTwo     | NULL   |  |        |        |                  |                    |                  |                     |                 |                     |              |      |              |      |

## RIGHT [OUTER] JOIN

| Запросы          | <pre>SELECT `r_name`, `c_name` FROM `rooms` RIGHT JOIN `computers` ON `r_id`=`c_room`</pre><br><pre>SELECT `r_name`, `c_name` FROM `rooms` RIGH OUTER JOIN `computers` ON `r_id`=`c_room`</pre>  |        |        |                  |                    |                  |                     |                 |                     |      |             |      |             |
|------------------|--|--------|--------|------------------|--------------------|------------------|---------------------|-----------------|---------------------|------|-------------|------|-------------|
| Результат        | <table><thead><tr><th>r_name</th><th>c_name</th></tr></thead><tbody><tr><td>RoomWithTwoComps</td><td>FirstCompInRoomOne</td></tr><tr><td>RoomWithTwoComps</td><td>SecondCompInRoomOne</td></tr><tr><td>RoomWithOneComp</td><td>SingleCompInRoomTwo</td></tr><tr><td>NULL</td><td>FreeCompOne</td></tr><tr><td>NULL</td><td>FreeCompTwo</td></tr></tbody></table> | r_name | c_name | RoomWithTwoComps | FirstCompInRoomOne | RoomWithTwoComps | SecondCompInRoomOne | RoomWithOneComp | SingleCompInRoomTwo | NULL | FreeCompOne | NULL | FreeCompTwo |
| r_name           | c_name   |        |        |                  |                    |                  |                     |                 |                     |      |             |      |             |
| RoomWithTwoComps | FirstCompInRoomOne   |        |        |                  |                    |                  |                     |                 |                     |      |             |      |             |
| RoomWithTwoComps | SecondCompInRoomOne  |        |        |                  |                    |                  |                     |                 |                     |      |             |      |             |
| RoomWithOneComp  | SingleCompInRoomTwo  |        |        |                  |                    |                  |                     |                 |                     |      |             |      |             |
| NULL             | FreeCompOne  |        |        |                  |                    |                  |                     |                 |                     |      |             |      |             |
| NULL             | FreeCompTwo  |        |        |                  |                    |                  |                     |                 |                     |      |             |      |             |

## 6.4.5. ВСЕ разновидности JOIN

```

table_references:
  table_reference [, table_reference] ...
table_reference:
  table_factor
  | join_table

table_factor:
  tbl_name [[AS] alias] [index_hint_list]
  | table_subquery [AS] alias
  | ( table_references )
  | { OJ table_reference LEFT OUTER JOIN table_reference
    ON conditional_expr }

join_table:
  table_reference [INNER | CROSS] JOIN table_factor [join_condition]
  | table_reference STRAIGHT_JOIN table_factor
  | table_reference STRAIGHT_JOIN table_factor ON conditional_expr
  | table_reference {LEFT|RIGHT} [OUTER] JOIN table_reference
  join_condition
  | table_reference NATURAL [{LEFT|RIGHT} [OUTER]] JOIN table_factor

join_condition:
  ON conditional_expr
  | USING (column_list)

index_hint_list:
  index_hint [, index_hint] ...

index_hint:
  USE (INDEX|KEY)
  | FOR {JOIN|ORDER BY|GROUP BY} [(index_list)]
  | IGNORE (INDEX|KEY)
  | FOR {JOIN|ORDER BY|GROUP BY} [(index_list)]
  | FORCE (INDEX|KEY)
  | FOR {JOIN|ORDER BY|GROUP BY} [(index_list)]

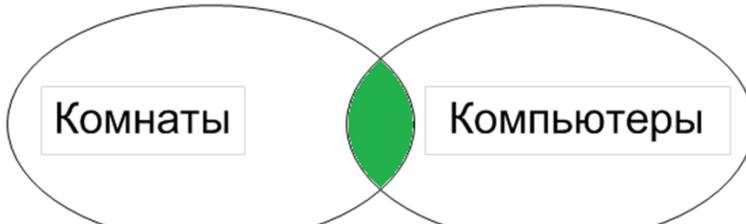
index_list:
  index_name [, index_name] ...

```

Теперь рассмотрим подробно, какие бывают JOIN'ы.

### {JOIN, INNER JOIN, CROSS JOIN} + USING или ON

В MySQL JOIN, INNER JOIN и CROSS JOIN – эквивалентны – обеспечивается поиск совпадений по всем таблицам по указанным полям.

| Запрос                            | <code>SELECT * FROM `rooms` JOIN `computers` ON<br/>`r_id` = `c_room`</code>  |      |        |                     |        |        |   |                  |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |
|-----------------------------------|---|------|--------|---------------------|--------|--------|---|------------------|---|---|--------------------|---|------------------|---|---|---------------------|---|-----------------|---|---|---------------------|
| Результат                         | <table border="1"> <thead> <tr> <th>r_id</th><th>r_name</th><th>c_id</th><th>c_room</th><th>c_name</th></tr> </thead> <tbody> <tr> <td>1</td><td>RoomWithTwoComps</td><td>1</td><td>1</td><td>FirstCompInRoomOne</td></tr> <tr> <td>1</td><td>RoomWithTwoComps</td><td>2</td><td>1</td><td>SecondCompInRoomOne</td></tr> <tr> <td>2</td><td>RoomWithOneComp</td><td>3</td><td>2</td><td>SingleCompInRoomTwo</td></tr> </tbody> </table> | r_id | r_name | c_id                | c_room | c_name | 1 | RoomWithTwoComps | 1 | 1 | FirstCompInRoomOne | 1 | RoomWithTwoComps | 2 | 1 | SecondCompInRoomOne | 2 | RoomWithOneComp | 3 | 2 | SingleCompInRoomTwo |
| r_id                              | r_name  | c_id | c_room | c_name              |        |        |   |                  |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |
| 1                                 | RoomWithTwoComps  | 1    | 1      | FirstCompInRoomOne  |        |        |   |                  |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |
| 1                                 | RoomWithTwoComps  | 2    | 1      | SecondCompInRoomOne |        |        |   |                  |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |
| 2                                 | RoomWithOneComp   | 3    | 2      | SingleCompInRoomTwo |        |        |   |                  |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |
| Пояснение с помощью кругов Эйлера |   |      |        |                     |        |        |   |                  |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |

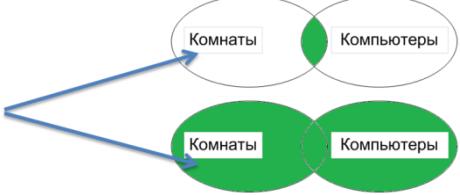
## {JOIN, INNER JOIN, CROSS JOIN} БЕЗ USING или ON; «,»

В случае отсутствия условия объединения (ON, USING) JOIN, INNER JOIN и CROSS JOIN и «,» эквивалентны – обеспечивается поиск всех комбинаций всех записей всех таблиц.

| Запрос                            | SELECT * FROM `rooms` JOIN `computers`   |      |        |                         |        |        |   |                  |   |   |                        |   |                 |   |   |                        |   |                |   |   |                        |   |                |   |   |                        |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |
|-----------------------------------|--|------|--------|-------------------------|--------|--------|---|------------------|---|---|------------------------|---|-----------------|---|---|------------------------|---|----------------|---|---|------------------------|---|----------------|---|---|------------------------|---|------------------|---|---|-------------------------|---|-----------------|---|---|-------------------------|---|----------------|---|---|-------------------------|---|----------------|---|---|-------------------------|---|------------------|---|---|-------------------------|---|-----------------|---|---|-------------------------|---|----------------|---|---|-------------------------|---|----------------|---|---|-------------------------|---|------------------|---|------|---------------|---|-----------------|---|------|---------------|---|----------------|---|------|---------------|---|----------------|---|------|---------------|---|------------------|---|------|---------------|---|-----------------|---|------|---------------|---|----------------|---|------|---------------|---|----------------|---|------|---------------|
| Результат                         | <table border="1"> <thead> <tr> <th>r_id</th><th>r_name</th><th>c_id</th><th>c_room</th><th>c_name</th></tr> </thead> <tbody> <tr><td>1</td><td>RoomWithTwoComps</td><td>1</td><td>1</td><td>First Comp In Room One</td></tr> <tr><td>2</td><td>RoomWithOneComp</td><td>1</td><td>1</td><td>First Comp In Room One</td></tr> <tr><td>3</td><td>Empty Room One</td><td>1</td><td>1</td><td>First Comp In Room One</td></tr> <tr><td>4</td><td>Empty Room Two</td><td>1</td><td>1</td><td>First Comp In Room One</td></tr> <tr><td>1</td><td>RoomWithTwoComps</td><td>2</td><td>1</td><td>Second Comp In Room One</td></tr> <tr><td>2</td><td>RoomWithOneComp</td><td>2</td><td>1</td><td>Second Comp In Room One</td></tr> <tr><td>3</td><td>Empty Room One</td><td>2</td><td>1</td><td>Second Comp In Room One</td></tr> <tr><td>4</td><td>Empty Room Two</td><td>2</td><td>1</td><td>Second Comp In Room One</td></tr> <tr><td>1</td><td>RoomWithTwoComps</td><td>3</td><td>2</td><td>Single Comp In Room Two</td></tr> <tr><td>2</td><td>RoomWithOneComp</td><td>3</td><td>2</td><td>Single Comp In Room Two</td></tr> <tr><td>3</td><td>Empty Room One</td><td>3</td><td>2</td><td>Single Comp In Room Two</td></tr> <tr><td>4</td><td>Empty Room Two</td><td>3</td><td>2</td><td>Single Comp In Room Two</td></tr> <tr><td>1</td><td>RoomWithTwoComps</td><td>4</td><td>NULL</td><td>Free Comp One</td></tr> <tr><td>2</td><td>RoomWithOneComp</td><td>4</td><td>NULL</td><td>Free Comp One</td></tr> <tr><td>3</td><td>Empty Room One</td><td>4</td><td>NULL</td><td>Free Comp One</td></tr> <tr><td>4</td><td>Empty Room Two</td><td>4</td><td>NULL</td><td>Free Comp One</td></tr> <tr><td>1</td><td>RoomWithTwoComps</td><td>5</td><td>NULL</td><td>Free Comp Two</td></tr> <tr><td>2</td><td>RoomWithOneComp</td><td>5</td><td>NULL</td><td>Free Comp Two</td></tr> <tr><td>3</td><td>Empty Room One</td><td>5</td><td>NULL</td><td>Free Comp Two</td></tr> <tr><td>4</td><td>Empty Room Two</td><td>5</td><td>NULL</td><td>Free Comp Two</td></tr> </tbody> </table> | r_id | r_name | c_id                    | c_room | c_name | 1 | RoomWithTwoComps | 1 | 1 | First Comp In Room One | 2 | RoomWithOneComp | 1 | 1 | First Comp In Room One | 3 | Empty Room One | 1 | 1 | First Comp In Room One | 4 | Empty Room Two | 1 | 1 | First Comp In Room One | 1 | RoomWithTwoComps | 2 | 1 | Second Comp In Room One | 2 | RoomWithOneComp | 2 | 1 | Second Comp In Room One | 3 | Empty Room One | 2 | 1 | Second Comp In Room One | 4 | Empty Room Two | 2 | 1 | Second Comp In Room One | 1 | RoomWithTwoComps | 3 | 2 | Single Comp In Room Two | 2 | RoomWithOneComp | 3 | 2 | Single Comp In Room Two | 3 | Empty Room One | 3 | 2 | Single Comp In Room Two | 4 | Empty Room Two | 3 | 2 | Single Comp In Room Two | 1 | RoomWithTwoComps | 4 | NULL | Free Comp One | 2 | RoomWithOneComp | 4 | NULL | Free Comp One | 3 | Empty Room One | 4 | NULL | Free Comp One | 4 | Empty Room Two | 4 | NULL | Free Comp One | 1 | RoomWithTwoComps | 5 | NULL | Free Comp Two | 2 | RoomWithOneComp | 5 | NULL | Free Comp Two | 3 | Empty Room One | 5 | NULL | Free Comp Two | 4 | Empty Room Two | 5 | NULL | Free Comp Two |
| r_id                              | r_name   | c_id | c_room | c_name                  |        |        |   |                  |   |   |                        |   |                 |   |   |                        |   |                |   |   |                        |   |                |   |   |                        |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |
| 1                                 | RoomWithTwoComps   | 1    | 1      | First Comp In Room One  |        |        |   |                  |   |   |                        |   |                 |   |   |                        |   |                |   |   |                        |   |                |   |   |                        |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |
| 2                                 | RoomWithOneComp  | 1    | 1      | First Comp In Room One  |        |        |   |                  |   |   |                        |   |                 |   |   |                        |   |                |   |   |                        |   |                |   |   |                        |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |
| 3                                 | Empty Room One   | 1    | 1      | First Comp In Room One  |        |        |   |                  |   |   |                        |   |                 |   |   |                        |   |                |   |   |                        |   |                |   |   |                        |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |
| 4                                 | Empty Room Two   | 1    | 1      | First Comp In Room One  |        |        |   |                  |   |   |                        |   |                 |   |   |                        |   |                |   |   |                        |   |                |   |   |                        |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |
| 1                                 | RoomWithTwoComps   | 2    | 1      | Second Comp In Room One |        |        |   |                  |   |   |                        |   |                 |   |   |                        |   |                |   |   |                        |   |                |   |   |                        |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |
| 2                                 | RoomWithOneComp  | 2    | 1      | Second Comp In Room One |        |        |   |                  |   |   |                        |   |                 |   |   |                        |   |                |   |   |                        |   |                |   |   |                        |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |
| 3                                 | Empty Room One   | 2    | 1      | Second Comp In Room One |        |        |   |                  |   |   |                        |   |                 |   |   |                        |   |                |   |   |                        |   |                |   |   |                        |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |
| 4                                 | Empty Room Two   | 2    | 1      | Second Comp In Room One |        |        |   |                  |   |   |                        |   |                 |   |   |                        |   |                |   |   |                        |   |                |   |   |                        |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |
| 1                                 | RoomWithTwoComps   | 3    | 2      | Single Comp In Room Two |        |        |   |                  |   |   |                        |   |                 |   |   |                        |   |                |   |   |                        |   |                |   |   |                        |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |
| 2                                 | RoomWithOneComp  | 3    | 2      | Single Comp In Room Two |        |        |   |                  |   |   |                        |   |                 |   |   |                        |   |                |   |   |                        |   |                |   |   |                        |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |
| 3                                 | Empty Room One   | 3    | 2      | Single Comp In Room Two |        |        |   |                  |   |   |                        |   |                 |   |   |                        |   |                |   |   |                        |   |                |   |   |                        |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |
| 4                                 | Empty Room Two   | 3    | 2      | Single Comp In Room Two |        |        |   |                  |   |   |                        |   |                 |   |   |                        |   |                |   |   |                        |   |                |   |   |                        |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |
| 1                                 | RoomWithTwoComps   | 4    | NULL   | Free Comp One           |        |        |   |                  |   |   |                        |   |                 |   |   |                        |   |                |   |   |                        |   |                |   |   |                        |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |
| 2                                 | RoomWithOneComp  | 4    | NULL   | Free Comp One           |        |        |   |                  |   |   |                        |   |                 |   |   |                        |   |                |   |   |                        |   |                |   |   |                        |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |
| 3                                 | Empty Room One   | 4    | NULL   | Free Comp One           |        |        |   |                  |   |   |                        |   |                 |   |   |                        |   |                |   |   |                        |   |                |   |   |                        |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |
| 4                                 | Empty Room Two   | 4    | NULL   | Free Comp One           |        |        |   |                  |   |   |                        |   |                 |   |   |                        |   |                |   |   |                        |   |                |   |   |                        |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |
| 1                                 | RoomWithTwoComps   | 5    | NULL   | Free Comp Two           |        |        |   |                  |   |   |                        |   |                 |   |   |                        |   |                |   |   |                        |   |                |   |   |                        |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |
| 2                                 | RoomWithOneComp  | 5    | NULL   | Free Comp Two           |        |        |   |                  |   |   |                        |   |                 |   |   |                        |   |                |   |   |                        |   |                |   |   |                        |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |
| 3                                 | Empty Room One   | 5    | NULL   | Free Comp Two           |        |        |   |                  |   |   |                        |   |                 |   |   |                        |   |                |   |   |                        |   |                |   |   |                        |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |
| 4                                 | Empty Room Two   | 5    | NULL   | Free Comp Two           |        |        |   |                  |   |   |                        |   |                 |   |   |                        |   |                |   |   |                        |   |                |   |   |                        |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |
| Пояснение с помощью кругов Эйлера |  |      |        |                         |        |        |   |                  |   |   |                        |   |                 |   |   |                        |   |                |   |   |                        |   |                |   |   |                        |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |   |                         |   |                 |   |   |                         |   |                |   |   |                         |   |                |   |   |                         |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |   |                  |   |      |               |   |                 |   |      |               |   |                |   |      |               |   |                |   |      |               |

## JOIN, STRAIGHT\_JOIN

**STRAIGHT\_JOIN** эквивалентен JOIN, но указывает, что левая таблица обязательно должна быть прочитана первой. Используется для указания оптимизатору запросов желаемого сценария работы.

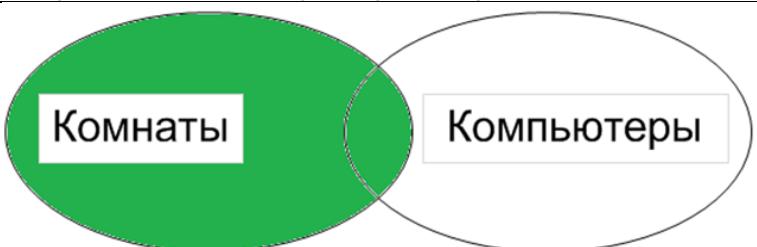
|                                   |   |
|-----------------------------------|---|
| Запросы                           | <pre>SELECT * FROM `rooms` STRAIGHT_JOIN `computers` ON `r_id`='c_room'</pre><br><pre>SELECT * FROM `rooms` STRAIGHT_JOIN `computers`</pre> |
| Пояснение с помощью кругов Эйлера | <p>Эта таблица будет прочитана первой.</p>               |

## LEFT [OUTER] JOIN, RIGHT [OUTER] JOIN

**ВАЖНО!** Слово OUTER можно не писать, т.к. в MySQL левые и правые объединения всегда выполняются как OUTER. Само слово введено для обеспечения совместимости.

### LEFT [OUTER] JOIN

При использовании LEFT [OUTER] JOIN происходит выборка всех строк ЛЕВОЙ таблицы + совпадений из правой таблицы. Если совпадений нет – на их место подставляются NULL.

| Запрос                            | <pre>SELECT * FROM `rooms` LEFT JOIN `computers` ON `r_id`='c_room'</pre>   |      |        |                     |        |        |   |                  |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |      |      |      |   |              |      |      |      |
|-----------------------------------|---|------|--------|---------------------|--------|--------|---|------------------|---|---|--------------------|---|------------------|---|---|---------------------|---|-----------------|---|---|---------------------|---|--------------|------|------|------|---|--------------|------|------|------|
| Результат                         | <table border="1"> <thead> <tr> <th>r_id</th> <th>r_name</th> <th>c_id</th> <th>c_room</th> <th>c_name</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>RoomWithTwoComps</td> <td>1</td> <td>1</td> <td>FirstCompInRoomOne</td> </tr> <tr> <td>1</td> <td>RoomWithTwoComps</td> <td>2</td> <td>1</td> <td>SecondCompInRoomOne</td> </tr> <tr> <td>2</td> <td>RoomWithOneComp</td> <td>3</td> <td>2</td> <td>SingleCompInRoomTwo</td> </tr> <tr> <td>3</td> <td>EmptyRoomOne</td> <td>NULL</td> <td>NULL</td> <td>NULL</td> </tr> <tr> <td>4</td> <td>EmptyRoomTwo</td> <td>NULL</td> <td>NULL</td> <td>NULL</td> </tr> </tbody> </table> | r_id | r_name | c_id                | c_room | c_name | 1 | RoomWithTwoComps | 1 | 1 | FirstCompInRoomOne | 1 | RoomWithTwoComps | 2 | 1 | SecondCompInRoomOne | 2 | RoomWithOneComp | 3 | 2 | SingleCompInRoomTwo | 3 | EmptyRoomOne | NULL | NULL | NULL | 4 | EmptyRoomTwo | NULL | NULL | NULL |
| r_id                              | r_name  | c_id | c_room | c_name              |        |        |   |                  |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |      |      |      |   |              |      |      |      |
| 1                                 | RoomWithTwoComps  | 1    | 1      | FirstCompInRoomOne  |        |        |   |                  |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |      |      |      |   |              |      |      |      |
| 1                                 | RoomWithTwoComps  | 2    | 1      | SecondCompInRoomOne |        |        |   |                  |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |      |      |      |   |              |      |      |      |
| 2                                 | RoomWithOneComp   | 3    | 2      | SingleCompInRoomTwo |        |        |   |                  |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |      |      |      |   |              |      |      |      |
| 3                                 | EmptyRoomOne  | NULL | NULL   | NULL                |        |        |   |                  |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |      |      |      |   |              |      |      |      |
| 4                                 | EmptyRoomTwo  | NULL | NULL   | NULL                |        |        |   |                  |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |      |      |      |   |              |      |      |      |
| Пояснение с помощью кругов Эйлера |   |      |        |                     |        |        |   |                  |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |      |      |      |   |              |      |      |      |

## RIGHT [OUTER] JOIN

При использовании RIGHT [OUTER] JOIN происходит выборка всех строк ПРАВОЙ таблицы + совпадений из левой таблицы. Если совпадений нет – на их место подставляются NULL.

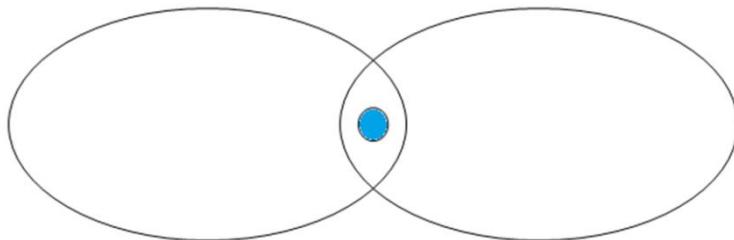
| Запрос                            | <code>SELECT * FROM `rooms` RIGHT JOIN `computers` ON `r_id` = `c_room`</code>  |      |        |                     |        |        |   |                  |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |      |      |   |      |             |      |      |   |      |             |
|-----------------------------------|---|------|--------|---------------------|--------|--------|---|------------------|---|---|--------------------|---|------------------|---|---|---------------------|---|-----------------|---|---|---------------------|------|------|---|------|-------------|------|------|---|------|-------------|
| Результат                         | <table border="1"> <thead> <tr> <th>r_id</th> <th>r_name</th> <th>c_id</th> <th>c_room</th> <th>c_name</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>RoomWithTwoComps</td> <td>1</td> <td>1</td> <td>FirstCompInRoomOne</td> </tr> <tr> <td>1</td> <td>RoomWithTwoComps</td> <td>2</td> <td>1</td> <td>SecondCompInRoomOne</td> </tr> <tr> <td>2</td> <td>RoomWithOneComp</td> <td>3</td> <td>2</td> <td>SingleCompInRoomTwo</td> </tr> <tr> <td>NULL</td> <td>NULL</td> <td>4</td> <td>NULL</td> <td>FreeCompOne</td> </tr> <tr> <td>NULL</td> <td>NULL</td> <td>5</td> <td>NULL</td> <td>FreeCompTwo</td> </tr> </tbody> </table> | r_id | r_name | c_id                | c_room | c_name | 1 | RoomWithTwoComps | 1 | 1 | FirstCompInRoomOne | 1 | RoomWithTwoComps | 2 | 1 | SecondCompInRoomOne | 2 | RoomWithOneComp | 3 | 2 | SingleCompInRoomTwo | NULL | NULL | 4 | NULL | FreeCompOne | NULL | NULL | 5 | NULL | FreeCompTwo |
| r_id                              | r_name  | c_id | c_room | c_name              |        |        |   |                  |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |      |      |   |      |             |      |      |   |      |             |
| 1                                 | RoomWithTwoComps  | 1    | 1      | FirstCompInRoomOne  |        |        |   |                  |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |      |      |   |      |             |      |      |   |      |             |
| 1                                 | RoomWithTwoComps  | 2    | 1      | SecondCompInRoomOne |        |        |   |                  |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |      |      |   |      |             |      |      |   |      |             |
| 2                                 | RoomWithOneComp   | 3    | 2      | SingleCompInRoomTwo |        |        |   |                  |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |      |      |   |      |             |      |      |   |      |             |
| NULL                              | NULL  | 4    | NULL   | FreeCompOne         |        |        |   |                  |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |      |      |   |      |             |      |      |   |      |             |
| NULL                              | NULL  | 5    | NULL   | FreeCompTwo         |        |        |   |                  |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |      |      |   |      |             |      |      |   |      |             |
| Пояснение с помощью кругов Эйлера |   |      |        |                     |        |        |   |                  |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |      |      |   |      |             |      |      |   |      |             |

## NATURAL [{LEFT|RIGHT} [OUTER]] JOIN

Редкий случай. Для объединения используются ВСЕ поля, существующие в обеих таблицах.

Фактически, это запрос на поиск записей-дубликатов в двух и более таблицах (в т.ч. Относящихся к разным базам данных).

Выглядеть результат будет так (запрос не приводим в силу экзотичности задачи):



## { OJ ... LEFT OUTER JOIN ... }

Существует только для совместимости с ODBC. Если вас это заинтересовало, милости просим сюда: <http://dev.mysql.com/doc/refman/5.5/en/join.html>

## 6.4.6. Условия объединения и дополнительные условия

```

table_references:
  table_reference [, table_reference] ...
  table_reference:
    table_factor
    | join_table

table_factor:
  tbl_name [AS] alias [index_hint_list]
  | table_subquery [AS] alias
  | ( table_references )
  | { OJ table_reference LEFT OUTER JOIN table_reference
    ON conditional_expr }

join_table:
  table_reference [INNER | CROSS] JOIN table_factor [join_condition]
  | table_reference STRAIGHT_JOIN table_factor
  | table_reference STRAIGHT_JOIN table_factor ON conditional_expr
  | table_reference (LEFT|RIGHT) [OUTER] JOIN table_reference join_condition
  | table_reference NATURAL [[LEFT|RIGHT] [OUTER]] JOIN table_factor

join_condition:
  ON conditional_expr
  | USING (column_list)

index_hint_list:
  index_hint [, index_hint] ...
  index_hint:
    USE (INDEX|KEY)
    | FOR (JOIN|ORDER BY|GROUP BY) ([index_list])
    | IGNORE (INDEX|KEY)
    | FOR (JOIN|ORDER BY|GROUP BY) (index_list)
    | FORCE (INDEX|KEY)
    | FOR (JOIN|ORDER BY|GROUP BY) (index_list)

index_list:
  index_name [, index_name] ...

```

Сейчас мы рассмотрим подробнее ON, USING и WHERE (HAVING).

**join\_condition:**  
 ON conditional\_expr  
 | USING (column\_list)

```

index_hint_list:
  index_hint [, index_hint] ...
  index_hint:
    USE (INDEX|KEY)
    | FOR (JOIN|ORDER BY|GROUP BY) ([index_list])
    | IGNORE (INDEX|KEY)
    | FOR (JOIN|ORDER BY|GROUP BY) (index_list)
    | FORCE (INDEX|KEY)
    | FOR (JOIN|ORDER BY|GROUP BY) (index_list)

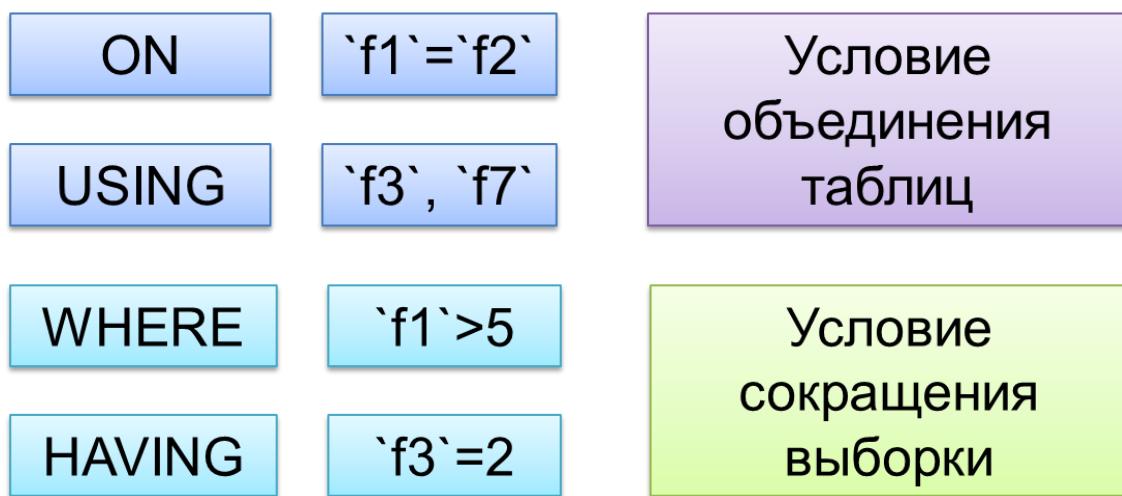
index_list:
  index_name [, index_name] ...

```

ON и USING используются для обозначения условий объединения (т.е. перечисляют поля, который нужно сравнивать между собой, причём в ON можно указывать разноимённые поля, а в USING указываются только поля, имеющие одинаковые имена в обеих таблицах).

WHERE (HAVING) используются для сокращения количества записей, попадающих в финальную выборку.

Ещё раз, графически ☺:



Пример: показать все компьютеры, стоящие в комнате 'RoomWithTwoComps'.

|           |   |
|-----------|---|
| Запрос    | <p>Условие объединения</p> <p>Условие сокращения выборки</p> <pre>SELECT `c_name` FROM `rooms` JOIN `computers`<br/>ON `r_id`='c_room'<br/>WHERE `r_name` = 'RoomWithTwoComps'</pre> <p>С использованием USING ситуация аналогичная, но для применения USING нужно, чтобы в таблицах были одноимённые поля. Это мы увидим чуть позже.</p> |
| Результат | <p>c_name</p> <p>FirstComplInRoomOne</p> <p>SecondComplInRoomOne</p>  |

С использованием USING ситуация аналогичная, но для применения USING нужно, чтобы в таблицах были одноимённые поля. Это мы увидим чуть позже.

### 6.4.7. Все виды JOIN в примерах

Сейчас мы рассмотрим, как ведут себя все возможные варианты JOIN в MySQL. Для проверки будем использовать уже знакомые нам таблицы `rooms` и `computers`. Только `computers` переименуем в `comps` и сделаем внешний ключ одноимённым первичному родительской таблицы (`r\_id`), чтобы можно было использовать USING.

| Таблица `rooms` |                  | Таблица `comps` |      |                     |
|-----------------|------------------|-----------------|------|---------------------|
| r_id            | r_name           | c_id            | r_id | c_name              |
| 1               | RoomWithTwoComps | 1               | 1    | FirstCompInRoomOne  |
| 2               | RoomWithOneComp  | 2               | 1    | SecondCompInRoomOne |
| 3               | EmptyRoomOne     | 3               | 2    | SingleCompInRoomTwo |
| 4               | EmptyRoomTwo     | 4               | NULL | FreeCompOne         |
|                 |                  | 5               | NULL | FreeCompTwo         |

#### Первая группа эквивалентных запросов

Результат: полный набор всех возможных комбинаций записей двух таблиц (фактически, ответ на вопрос «как можно разместить компьютеры по комнатам»)

```
SELECT * FROM `rooms` JOIN `comps`
SELECT * FROM `rooms` FULL JOIN `comps`
SELECT * FROM `rooms`, `comps`
SELECT * FROM `rooms` INNER JOIN `comps`
SELECT * FROM `rooms` CROSS JOIN `comps`
```

В руководстве по MySQL 5.x этого нет (было раньше), но (пока?) работает.

Вот, как это выглядит...

| Запрос    | SELECT * FROM `rooms` JOIN `comps`  |      |        |                     |      |        |   |                  |   |   |                    |   |                 |   |   |                    |   |              |   |   |                    |   |              |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |
|-----------|---|------|--------|---------------------|------|--------|---|------------------|---|---|--------------------|---|-----------------|---|---|--------------------|---|--------------|---|---|--------------------|---|--------------|---|---|--------------------|---|------------------|---|---|---------------------|---|-----------------|---|---|---------------------|---|--------------|---|---|---------------------|---|--------------|---|---|---------------------|---|------------------|---|---|---------------------|---|-----------------|---|---|---------------------|---|--------------|---|---|---------------------|---|--------------|---|---|---------------------|---|------------------|---|------|-------------|---|-----------------|---|------|-------------|---|--------------|---|------|-------------|---|--------------|---|------|-------------|---|------------------|---|------|-------------|---|-----------------|---|------|-------------|---|--------------|---|------|-------------|---|--------------|---|------|-------------|
| Результат | <table border="1"> <thead> <tr> <th>r_id</th> <th>r_name</th> <th>c_id</th> <th>r_id</th> <th>c_name</th> </tr> </thead> <tbody> <tr><td>1</td><td>RoomWithTwoComps</td><td>1</td><td>1</td><td>FirstCompInRoomOne</td></tr> <tr><td>2</td><td>RoomWithOneComp</td><td>1</td><td>1</td><td>FirstCompInRoomOne</td></tr> <tr><td>3</td><td>EmptyRoomOne</td><td>1</td><td>1</td><td>FirstCompInRoomOne</td></tr> <tr><td>4</td><td>EmptyRoomTwo</td><td>1</td><td>1</td><td>FirstCompInRoomOne</td></tr> <tr><td>1</td><td>RoomWithTwoComps</td><td>2</td><td>1</td><td>SecondCompInRoomOne</td></tr> <tr><td>2</td><td>RoomWithOneComp</td><td>2</td><td>1</td><td>SecondCompInRoomOne</td></tr> <tr><td>3</td><td>EmptyRoomOne</td><td>2</td><td>1</td><td>SecondCompInRoomOne</td></tr> <tr><td>4</td><td>EmptyRoomTwo</td><td>2</td><td>1</td><td>SecondCompInRoomOne</td></tr> <tr><td>1</td><td>RoomWithTwoComps</td><td>3</td><td>2</td><td>SingleCompInRoomTwo</td></tr> <tr><td>2</td><td>RoomWithOneComp</td><td>3</td><td>2</td><td>SingleCompInRoomTwo</td></tr> <tr><td>3</td><td>EmptyRoomOne</td><td>3</td><td>2</td><td>SingleCompInRoomTwo</td></tr> <tr><td>4</td><td>EmptyRoomTwo</td><td>3</td><td>2</td><td>SingleCompInRoomTwo</td></tr> <tr><td>1</td><td>RoomWithTwoComps</td><td>4</td><td>NULL</td><td>FreeCompOne</td></tr> <tr><td>2</td><td>RoomWithOneComp</td><td>4</td><td>NULL</td><td>FreeCompOne</td></tr> <tr><td>3</td><td>EmptyRoomOne</td><td>4</td><td>NULL</td><td>FreeCompOne</td></tr> <tr><td>4</td><td>EmptyRoomTwo</td><td>4</td><td>NULL</td><td>FreeCompOne</td></tr> <tr><td>1</td><td>RoomWithTwoComps</td><td>5</td><td>NULL</td><td>FreeCompTwo</td></tr> <tr><td>2</td><td>RoomWithOneComp</td><td>5</td><td>NULL</td><td>FreeCompTwo</td></tr> <tr><td>3</td><td>EmptyRoomOne</td><td>5</td><td>NULL</td><td>FreeCompTwo</td></tr> <tr><td>4</td><td>EmptyRoomTwo</td><td>5</td><td>NULL</td><td>FreeCompTwo</td></tr> </tbody> </table> | r_id | r_name | c_id                | r_id | c_name | 1 | RoomWithTwoComps | 1 | 1 | FirstCompInRoomOne | 2 | RoomWithOneComp | 1 | 1 | FirstCompInRoomOne | 3 | EmptyRoomOne | 1 | 1 | FirstCompInRoomOne | 4 | EmptyRoomTwo | 1 | 1 | FirstCompInRoomOne | 1 | RoomWithTwoComps | 2 | 1 | SecondCompInRoomOne | 2 | RoomWithOneComp | 2 | 1 | SecondCompInRoomOne | 3 | EmptyRoomOne | 2 | 1 | SecondCompInRoomOne | 4 | EmptyRoomTwo | 2 | 1 | SecondCompInRoomOne | 1 | RoomWithTwoComps | 3 | 2 | SingleCompInRoomTwo | 2 | RoomWithOneComp | 3 | 2 | SingleCompInRoomTwo | 3 | EmptyRoomOne | 3 | 2 | SingleCompInRoomTwo | 4 | EmptyRoomTwo | 3 | 2 | SingleCompInRoomTwo | 1 | RoomWithTwoComps | 4 | NULL | FreeCompOne | 2 | RoomWithOneComp | 4 | NULL | FreeCompOne | 3 | EmptyRoomOne | 4 | NULL | FreeCompOne | 4 | EmptyRoomTwo | 4 | NULL | FreeCompOne | 1 | RoomWithTwoComps | 5 | NULL | FreeCompTwo | 2 | RoomWithOneComp | 5 | NULL | FreeCompTwo | 3 | EmptyRoomOne | 5 | NULL | FreeCompTwo | 4 | EmptyRoomTwo | 5 | NULL | FreeCompTwo |
| r_id      | r_name  | c_id | r_id   | c_name              |      |        |   |                  |   |   |                    |   |                 |   |   |                    |   |              |   |   |                    |   |              |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |
| 1         | RoomWithTwoComps  | 1    | 1      | FirstCompInRoomOne  |      |        |   |                  |   |   |                    |   |                 |   |   |                    |   |              |   |   |                    |   |              |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |
| 2         | RoomWithOneComp   | 1    | 1      | FirstCompInRoomOne  |      |        |   |                  |   |   |                    |   |                 |   |   |                    |   |              |   |   |                    |   |              |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |
| 3         | EmptyRoomOne  | 1    | 1      | FirstCompInRoomOne  |      |        |   |                  |   |   |                    |   |                 |   |   |                    |   |              |   |   |                    |   |              |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |
| 4         | EmptyRoomTwo  | 1    | 1      | FirstCompInRoomOne  |      |        |   |                  |   |   |                    |   |                 |   |   |                    |   |              |   |   |                    |   |              |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |
| 1         | RoomWithTwoComps  | 2    | 1      | SecondCompInRoomOne |      |        |   |                  |   |   |                    |   |                 |   |   |                    |   |              |   |   |                    |   |              |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |
| 2         | RoomWithOneComp   | 2    | 1      | SecondCompInRoomOne |      |        |   |                  |   |   |                    |   |                 |   |   |                    |   |              |   |   |                    |   |              |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |
| 3         | EmptyRoomOne  | 2    | 1      | SecondCompInRoomOne |      |        |   |                  |   |   |                    |   |                 |   |   |                    |   |              |   |   |                    |   |              |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |
| 4         | EmptyRoomTwo  | 2    | 1      | SecondCompInRoomOne |      |        |   |                  |   |   |                    |   |                 |   |   |                    |   |              |   |   |                    |   |              |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |
| 1         | RoomWithTwoComps  | 3    | 2      | SingleCompInRoomTwo |      |        |   |                  |   |   |                    |   |                 |   |   |                    |   |              |   |   |                    |   |              |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |
| 2         | RoomWithOneComp   | 3    | 2      | SingleCompInRoomTwo |      |        |   |                  |   |   |                    |   |                 |   |   |                    |   |              |   |   |                    |   |              |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |
| 3         | EmptyRoomOne  | 3    | 2      | SingleCompInRoomTwo |      |        |   |                  |   |   |                    |   |                 |   |   |                    |   |              |   |   |                    |   |              |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |
| 4         | EmptyRoomTwo  | 3    | 2      | SingleCompInRoomTwo |      |        |   |                  |   |   |                    |   |                 |   |   |                    |   |              |   |   |                    |   |              |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |
| 1         | RoomWithTwoComps  | 4    | NULL   | FreeCompOne         |      |        |   |                  |   |   |                    |   |                 |   |   |                    |   |              |   |   |                    |   |              |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |
| 2         | RoomWithOneComp   | 4    | NULL   | FreeCompOne         |      |        |   |                  |   |   |                    |   |                 |   |   |                    |   |              |   |   |                    |   |              |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |
| 3         | EmptyRoomOne  | 4    | NULL   | FreeCompOne         |      |        |   |                  |   |   |                    |   |                 |   |   |                    |   |              |   |   |                    |   |              |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |
| 4         | EmptyRoomTwo  | 4    | NULL   | FreeCompOne         |      |        |   |                  |   |   |                    |   |                 |   |   |                    |   |              |   |   |                    |   |              |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |
| 1         | RoomWithTwoComps  | 5    | NULL   | FreeCompTwo         |      |        |   |                  |   |   |                    |   |                 |   |   |                    |   |              |   |   |                    |   |              |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |
| 2         | RoomWithOneComp   | 5    | NULL   | FreeCompTwo         |      |        |   |                  |   |   |                    |   |                 |   |   |                    |   |              |   |   |                    |   |              |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |
| 3         | EmptyRoomOne  | 5    | NULL   | FreeCompTwo         |      |        |   |                  |   |   |                    |   |                 |   |   |                    |   |              |   |   |                    |   |              |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |
| 4         | EmptyRoomTwo  | 5    | NULL   | FreeCompTwo         |      |        |   |                  |   |   |                    |   |                 |   |   |                    |   |              |   |   |                    |   |              |   |   |                    |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |   |                     |   |                 |   |   |                     |   |              |   |   |                     |   |              |   |   |                     |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |   |                  |   |      |             |   |                 |   |      |             |   |              |   |      |             |   |              |   |      |             |

Обратите внимание: в результирующей выборке два одноимённых поля (`r\_id`). Этого можно избежать, если вместо «SELECT \*» указывать конкретные имена полей, и одноимённые «переименовывать» через AS.

## Лирическое отступление 😊

Запросы вида

```
SELECT * FROM `table1`, `table2`, ... , `tableN`
```

очень удобно использовать для быстрого получения всех возможных комбинаций множества значений, например, в автоматизированном тестировании.

Достаточно создать под каждый параметр свою таблицу, занести туда все интересующие нас значения, а потом «проджойнить» получившиеся таблицы.

## Вторая группа эквивалентных запросов

Результат: список комнат, в которых стоят компьютеры (что можно понимать и как «список компьютеров, которые стоят в комнатах»). Главное – сюда НЕ попадут пустые комнаты и «бесхозные» компьютеры.

```
SELECT * FROM `rooms` JOIN `comps` USING (`r_id`)
SELECT * FROM `rooms` FULL JOIN `comps` USING (`r_id`)
SELECT * FROM `rooms` INNER JOIN `comps` USING (`r_id`)
SELECT * FROM `rooms` CROSS JOIN `comps` USING (`r_id`)
SELECT * FROM `rooms` , `comps` USING (`r_id`)
```

USING нельзя использовать с «,»

Вот, как это выглядит...

| Запрос    | SELECT * FROM `rooms` JOIN `comps` USING (`r_id`) |                  |      |                      |
|-----------|---|------------------|------|----------------------|
| Результат | r_id  | r_name           | c_id | c_name               |
|           | 1   | RoomWithTwoComps | 1    | FirstComplInRoomOne  |
|           | 1   | RoomWithTwoComps | 2    | SecondComplInRoomOne |
|           | 2   | RoomWithOneComp  | 3    | SingleComplInRoomTwo |

В этом примере поле `r\_id` не дублируется.

## Третья группа эквивалентных запросов

Результат: список ВСЕХ комнат: в которых стоят компьютеры (с указанием компьютеров) и пустых (NULL вместо компьютера). Компьютеры, не поставленные в комнату, в этот результат не попадают.

```
SELECT * FROM `rooms` LEFT JOIN `comps` USING (`r_id`)
SELECT * FROM `rooms` LEFT OUTER JOIN `comps` USING (`r_id`)
SELECT * FROM `rooms` LEFT JOIN `r_id` 
SELECT * FROM `rooms` LEFT OUTER JOIN `r_id`
```

Необходимо указывать условие объединения!

Вот, как это выглядит...

| Запрос    | SELECT * FROM `rooms` LEFT JOIN `comps` USING (`r_id`) |                  |      |                      |
|-----------|--|------------------|------|----------------------|
| Результат | r_id   | r_name           | c_id | c_name               |
|           | 1  | RoomWithTwoComps | 1    | FirstComplInRoomOne  |
|           | 1  | RoomWithTwoComps | 2    | SecondComplInRoomOne |
|           | 2  | RoomWithOneComp  | 3    | SingleComplInRoomTwo |
|           | 3  | EmptyRoomOne     | NULL | NULL                 |
|           | 4  | EmptyRoomTwo     | NULL | NULL                 |

### Четвёртая группа эквивалентных запросов

Результат: список ВСЕХ компьютеров: которые стоят в комнатах (с указанием комнат) и «бесхозных» (NULL вместо комнаты). Комнаты, в которых нет компьютеров, в этот результат не попадают.

```
SELECT * FROM `rooms` RIGHT JOIN `comps` USING (`r_id`)
SELECT * FROM `rooms` RIGHT OUTER JOIN `comps` USING (`r_id`)
SELECT * FROM `rooms` RIGHT JOIN `r_id`
SELECT * FROM `rooms` RIGHT OUTER JOIN `r_id`
```

Необходимо указывать условие объединения!

Вот, как это выглядит...

| Запрос    | SELECT * FROM `rooms` RIGHT JOIN `comps` USING (`r_id`) |      |                      |                  |
|-----------|---|------|----------------------|------------------|
| Результат | r_id  | c_id | c_name               | r_name           |
|           | 1   | 1    | FirstComplInRoomOne  | RoomWithTwoComps |
|           | 1   | 2    | SecondComplInRoomOne | RoomWithTwoComps |
|           | 2   | 3    | SingleComplInRoomTwo | RoomWithOneComp  |
|           | NULL  | 4    | FreeCompOne          | NULL             |
|           | NULL  | 5    | FreeCompTwo          | NULL             |

### 6.4.8. JOIN'ы и NULL'ы

**Q:** Когда и где NULL'ы (реально содержащиеся в полях, по которым происходит объединение) сохраняются и пропадают при использовании JOIN?

**A:** Пропадают при JOIN, FULL JOIN, INNER JOIN, CROSS JOIN, “,”. Сохраняются при LEFT [OUTER] JOIN, RIGHT [OUTER] JOIN.

**Q:** Не понятно ☹.

**A:** Жаль ☹. Т.е. сейчас рассмотрим пример.

Пусть у нас есть таблицы `in` и `out`, фиксирующие приход на работу и уход с работы сотрудника. Для полноты картины представим, что данные вносятся вручную, и нерадивый вахтёр часто забывает, как сотрудника зовут. Тогда мы получим, например, такое:

| Таблица `in` |             | Таблица `out` |             |
|--------------|-------------|---------------|-------------|
| <b>id</b>    | <b>name</b> | <b>id</b>     | <b>name</b> |
| 1            | Пупкин      | 1             | Иванов      |
| 2            | Сидоров     | 2             | NULL        |
| 3            | NULL        | 3             | Петров      |
| 4            | Петров      | 4             | Пупкин      |

Теперь напишем запросы, с помощью которых постараемся выяснить ситуацию вида

- (A) «пришёл и ушёл»;
- (Б) «(А) + пришёл, но не ушёл»;
- (В) «(А) + не приходил, но ушёл».



## «Пришёл и ушёл»

| Запросы   | <code>SELECT * FROM `in` JOIN `out` USING (`name`)</code>   |      |    |    |        |   |   |        |   |   |
|-----------|---|------|----|----|--------|---|---|--------|---|---|
|           | <code>SELECT * FROM `in` FULL JOIN `out` USING (`name`)</code>  |      |    |    |        |   |   |        |   |   |
|           | <code>SELECT * FROM `in` INNER JOIN `out` USING (`name`)</code>   |      |    |    |        |   |   |        |   |   |
|           | <code>SELECT * FROM `in` CROSS JOIN `out` USING (`name`)</code>   |      |    |    |        |   |   |        |   |   |
| Результат | <table border="1"> <thead> <tr> <th>name</th><th>id</th><th>id</th></tr> </thead> <tbody> <tr> <td>Петров</td><td>4</td><td>3</td></tr> <tr> <td>Пупкин</td><td>1</td><td>4</td></tr> </tbody> </table> | name | id | id | Петров | 4 | 3 | Пупкин | 1 | 4 |
| name      | id  | id   |    |    |        |   |   |        |   |   |
| Петров    | 4   | 3    |    |    |        |   |   |        |   |   |
| Пупкин    | 1   | 4    |    |    |        |   |   |        |   |   |

## «Пришёл и ушёл + пришёл, но не ушёл»

| Запросы   | <code>SELECT * FROM `in` LEFT JOIN `out` USING (`name`)</code>   |      |    |    |        |   |   |         |   |      |      |   |      |        |   |   |
|-----------|--|------|----|----|--------|---|---|---------|---|------|------|---|------|--------|---|---|
|           | <code>SELECT * FROM `in` LEFT OUTER JOIN `out` USING (`name`)</code>   |      |    |    |        |   |   |         |   |      |      |   |      |        |   |   |
| Результат | <table border="1"> <thead> <tr> <th>name</th><th>id</th><th>id</th></tr> </thead> <tbody> <tr> <td>Пупкин</td><td>1</td><td>4</td></tr> <tr> <td>Сидоров</td><td>2</td><td>NULL</td></tr> <tr> <td>NULL</td><td>3</td><td>NULL</td></tr> <tr> <td>Петров</td><td>4</td><td>3</td></tr> </tbody> </table> | name | id | id | Пупкин | 1 | 4 | Сидоров | 2 | NULL | NULL | 3 | NULL | Петров | 4 | 3 |
| name      | id   | id   |    |    |        |   |   |         |   |      |      |   |      |        |   |   |
| Пупкин    | 1  | 4    |    |    |        |   |   |         |   |      |      |   |      |        |   |   |
| Сидоров   | 2  | NULL |    |    |        |   |   |         |   |      |      |   |      |        |   |   |
| NULL      | 3  | NULL |    |    |        |   |   |         |   |      |      |   |      |        |   |   |
| Петров    | 4  | 3    |    |    |        |   |   |         |   |      |      |   |      |        |   |   |

## «Пришёл и ушёл + не приходил, но ушёл»

| Запросы   | <code>SELECT * FROM `in` RIGHT JOIN `out` USING (`name`)</code>   |      |    |    |        |   |      |      |   |      |        |   |   |        |   |   |
|-----------|---|------|----|----|--------|---|------|------|---|------|--------|---|---|--------|---|---|
|           | <code>SELECT * FROM `in` RIGHT OUTER JOIN `out` USING (`name`)</code>   |      |    |    |        |   |      |      |   |      |        |   |   |        |   |   |
| Результат | <table border="1"> <thead> <tr> <th>name</th><th>id</th><th>id</th></tr> </thead> <tbody> <tr> <td>Иванов</td><td>1</td><td>NULL</td></tr> <tr> <td>NULL</td><td>2</td><td>NULL</td></tr> <tr> <td>Петров</td><td>3</td><td>4</td></tr> <tr> <td>Пупкин</td><td>4</td><td>1</td></tr> </tbody> </table> | name | id | id | Иванов | 1 | NULL | NULL | 2 | NULL | Петров | 3 | 4 | Пупкин | 4 | 1 |
| name      | id  | id   |    |    |        |   |      |      |   |      |        |   |   |        |   |   |
| Иванов    | 1   | NULL |    |    |        |   |      |      |   |      |        |   |   |        |   |   |
| NULL      | 2   | NULL |    |    |        |   |      |      |   |      |        |   |   |        |   |   |
| Петров    | 3   | 4    |    |    |        |   |      |      |   |      |        |   |   |        |   |   |
| Пупкин    | 4   | 1    |    |    |        |   |      |      |   |      |        |   |   |        |   |   |

### 6.4.9. JOIN'ы и дублирование имён полей

**Q:** Во всех предыдущих примерах дублируется поле `id`. Это отвратительно! Как от этого избавиться!?

**A:** Использовать AS в запросе для переименования полей. Вот так:

| Запрос    | <pre>SELECT `in`.`name`, `in`.`id` AS `IN_in`, `out`.`id` AS `OUT_id` FROM `in` LEFT JOIN `out` USING (`name`)</pre>  |        |  |      |       |        |        |   |   |         |   |      |      |   |      |        |   |   |
|-----------|---|--------|--|------|-------|--------|--------|---|---|---------|---|------|------|---|------|--------|---|---|
| Результат | <table><thead><tr><th>name</th><th>IN_in</th><th>OUT_id</th></tr></thead><tbody><tr><td>Пупкин</td><td>1</td><td>4</td></tr><tr><td>Сидоров</td><td>2</td><td>NULL</td></tr><tr><td>NULL</td><td>3</td><td>NULL</td></tr><tr><td>Петров</td><td>4</td><td>3</td></tr></tbody></table> |        |  | name | IN_in | OUT_id | Пупкин | 1 | 4 | Сидоров | 2 | NULL | NULL | 3 | NULL | Петров | 4 | 3 |
| name      | IN_in   | OUT_id |  |      |       |        |        |   |   |         |   |      |      |   |      |        |   |   |
| Пупкин    | 1   | 4      |  |      |       |        |        |   |   |         |   |      |      |   |      |        |   |   |
| Сидоров   | 2   | NULL   |  |      |       |        |        |   |   |         |   |      |      |   |      |        |   |   |
| NULL      | 3   | NULL   |  |      |       |        |        |   |   |         |   |      |      |   |      |        |   |   |
| Петров    | 4   | 3      |  |      |       |        |        |   |   |         |   |      |      |   |      |        |   |   |

### 6.4.10. Нетривиальные случаи, вопросы и примеры

Итак, всё типичное с JOIN'ами мы рассмотрели. Теперь рассмотрим нетипичное.

#### Рекурсивные JOIN'ы

**Q:** Как сделать "рекурсивный JOIN"? Например, чтобы скопировать ветку дерева.

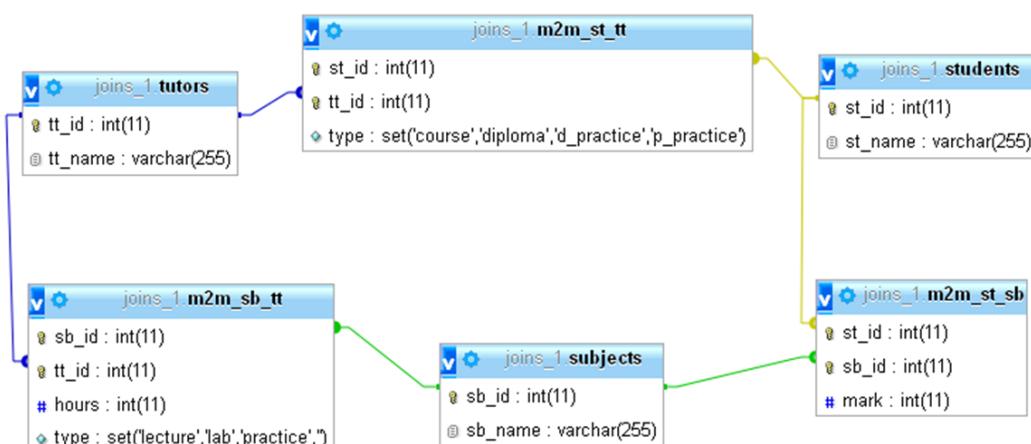
**A:** Никак ☹, т.к. MySQL не поддерживает рекурсивные JOIN'ы. Надо или написать хранимую процедуру, которая «выполнит рекурсию» в виде цикла, или написать рекурсивную хранимую процедуру (не функцию, т.к. они не поддерживаются).

Подробности: <http://stackoverflow.com/questions/3536538/mysql-does-not-support-recursive-functions-why-since-when>

См. также альтернативные способы хранения и обработки иерархических структур в реляционных БД: <http://www.slideshare.net/billkarwin/models-for-hierarchical-data>

#### Как исследование запросов влияет на модель БД

Представим классический пример части модели БД «Деканат». Эта часть описывает студентов, преподавателей и предметы.



Мы можем выполнить здесь несколько примитивных запросов в стиле «показать, кто у кого учится», «показать, кто что изучает», «показать, кто что ведёт».

Например, вот запрос, показывающий «какие студенты изучают какие предметы и какие у них оценки».

```

SELECT `st_name`, `sb_name`, `mark`
from `students`, `subjects`, `m2m_st_sb`
WHERE `students`.`st_id` = `m2m_st_sb`.`st_id`
AND `m2m_st_sb`.`sb_id` = `subjects`.`sb_id`
    
```

Или так (запросы эквивалентны):

```

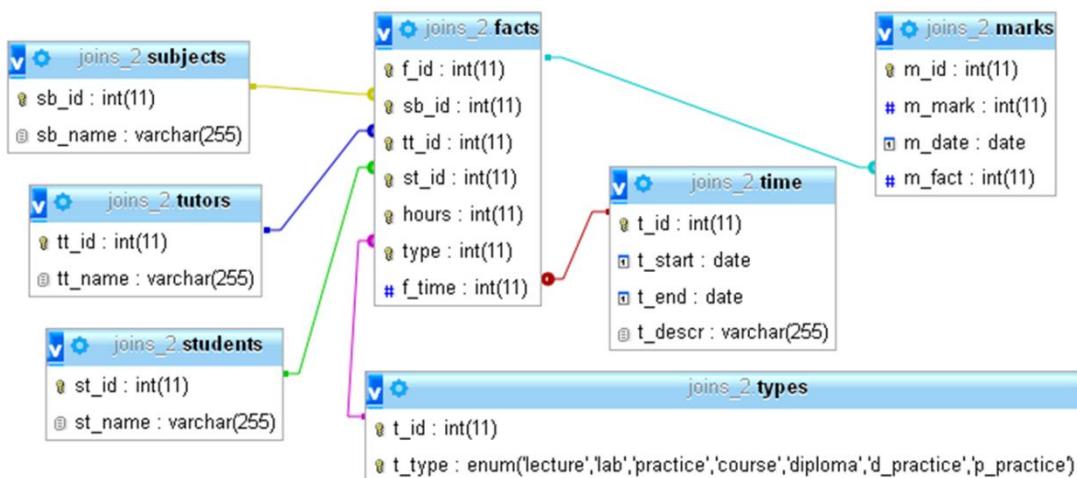
SELECT `st_name`, `sb_name`, `mark` from
`students` JOIN `m2m_st_sb` ON `students`.`st_id` = `m2m_st_sb`.`st_id` JOIN
`subjects` ON `m2m_st_sb`.`sb_id` = `subjects`.`sb_id`
    
```

Но! Как только возникает вопрос в стиле «Вывести среднее значение всех оценок, выставленных конкретным преподавателем», выясняется, что БД не содержит этой информации. Т.е. мы знаем, что такие-то преподаватели ведут такие-

то предметы у таких-то студентов, но кто конкретно выставлял оценку – выяснить нельзя.

Более того, анализ модели показывает её полное несоответствие реальности (как показать пересдачу, повторное обучение после отчисления и восстановления, как отличать курсовые (одна оценка) и лабораторные (много оценок) и т.д.)

Если модель переработать, получается вот так (да-да, эту модель можно ещё дорабатывать и дорабатывать):



Теперь ответ на вопрос «Как вывести среднее значение всех оценок, выставленных конкретным преподавателем?» выглядит так:

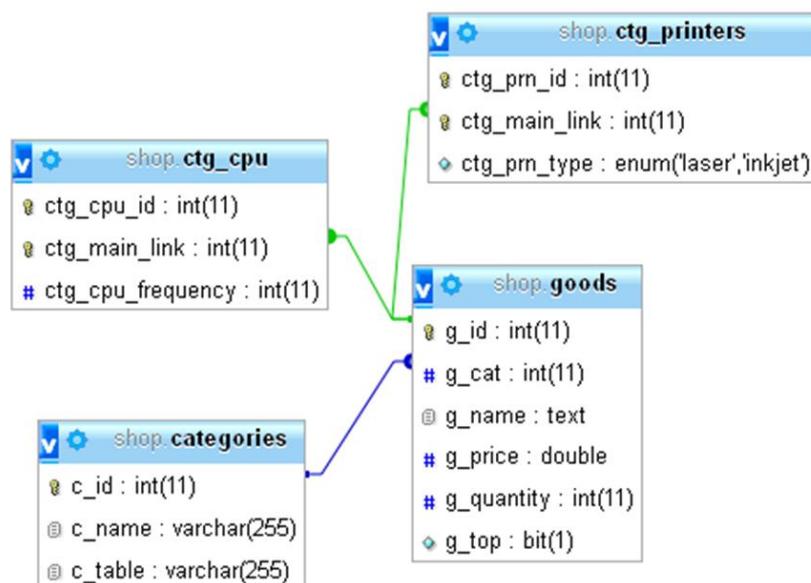
| Запрос      | <pre> SELECT `tutors`.`tt_name`, AVG(`marks`.`m_mark`) as `avg_mark` FROM `marks` JOIN `facts` ON `marks`.`m_fact` = `facts`.`f_id` JOIN `tutors` ON `facts`.`tt_id` = `tutors`.`tt_id` GROUP BY `tutors`.`tt_name`   </pre>  |         |          |           |        |           |        |           |        |             |        |             |        |
|-------------|---|---------|----------|-----------|--------|-----------|--------|-----------|--------|-------------|--------|-------------|--------|
| Результат   | <table border="1"> <thead> <tr> <th>tt_name</th> <th>avg_mark</th> </tr> </thead> <tbody> <tr> <td>TestTutor</td> <td>7.5000</td> </tr> <tr> <td>TT_Kind_A</td> <td>9.0000</td> </tr> <tr> <td>TT_Kind_B</td> <td>8.6667</td> </tr> <tr> <td>TT_Strict_A</td> <td>9.0000</td> </tr> <tr> <td>TT_Strict_B</td> <td>4.0000</td> </tr> </tbody> </table> | tt_name | avg_mark | TestTutor | 7.5000 | TT_Kind_A | 9.0000 | TT_Kind_B | 8.6667 | TT_Strict_A | 9.0000 | TT_Strict_B | 4.0000 |
| tt_name     | avg_mark  |         |          |           |        |           |        |           |        |             |        |             |        |
| TestTutor   | 7.5000  |         |          |           |        |           |        |           |        |             |        |             |        |
| TT_Kind_A   | 9.0000  |         |          |           |        |           |        |           |        |             |        |             |        |
| TT_Kind_B   | 8.6667  |         |          |           |        |           |        |           |        |             |        |             |        |
| TT_Strict_A | 9.0000  |         |          |           |        |           |        |           |        |             |        |             |        |
| TT_Strict_B | 4.0000  |         |          |           |        |           |        |           |        |             |        |             |        |

## Ещё более нетипичные модель БД и JOIN'ы

Представим, что модель БД интернет-магазина построена по следующей логике: есть единая таблица, в которой хранятся универсальные характеристики товаров (наименование, цена, количество и т.д.), а все уникальные характеристики хранятся в отдельных таблицах, каждая из которых описывает одну категорию товаров. Далее между «единой» и «дополнительными» таблицами проведены связи типа «один к одному».

Да, в реальности с такой моделью могут быть проблемы. Но если реализовать её вдумчиво и аккуратно, всё будет в порядке.

Итак, вот эта модель:



Наполним таблицы данными:

| Таблица        | Данные   |                   |               |                   |         |            |         |   |          |              |     |   |        |   |   |        |     |    |   |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |
|----------------|--|-------------------|---------------|-------------------|---------|------------|---------|---|----------|--------------|-----|---|--------|---|---|--------|-----|----|---|---|---|--------|-----|---|---|---|---|--------|-----|---|---|---|---|------------|-----|---|---|
| `categories`   | <table border="1"> <thead> <tr> <th>c_id</th><th>c_name</th><th>c_table</th></tr> </thead> <tbody> <tr> <td>1</td><td>CPU</td><td>ctg_cpu</td></tr> <tr> <td>2</td><td>Printers</td><td>ctg_printers</td></tr> </tbody> </table>   | c_id              | c_name        | c_table           | 1       | CPU        | ctg_cpu | 2 | Printers | ctg_printers |     |   |        |   |   |        |     |    |   |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |
| c_id           | c_name   | c_table           |               |                   |         |            |         |   |          |              |     |   |        |   |   |        |     |    |   |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |
| 1              | CPU  | ctg_cpu           |               |                   |         |            |         |   |          |              |     |   |        |   |   |        |     |    |   |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |
| 2              | Printers   | ctg_printers      |               |                   |         |            |         |   |          |              |     |   |        |   |   |        |     |    |   |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |
| `goods`        | <table border="1"> <thead> <tr> <th>g_id</th><th>g_cat</th><th>g_name</th><th>g_price</th><th>g_quantity</th><th>g_top</th></tr> </thead> <tbody> <tr> <td>1</td><td>1</td><td>i7</td><td>250</td><td>5</td><td>0</td></tr> <tr> <td>2</td><td>1</td><td>Athlon</td><td>250</td><td>10</td><td>1</td></tr> <tr> <td>3</td><td>2</td><td>HP1010</td><td>300</td><td>5</td><td>1</td></tr> <tr> <td>4</td><td>2</td><td>HP2020</td><td>400</td><td>7</td><td>0</td></tr> <tr> <td>5</td><td>2</td><td>Epson-N999</td><td>500</td><td>9</td><td>1</td></tr> </tbody> </table> | g_id              | g_cat         | g_name            | g_price | g_quantity | g_top   | 1 | 1        | i7           | 250 | 5 | 0      | 2 | 1 | Athlon | 250 | 10 | 1 | 3 | 2 | HP1010 | 300 | 5 | 1 | 4 | 2 | HP2020 | 400 | 7 | 0 | 5 | 2 | Epson-N999 | 500 | 9 | 1 |
| g_id           | g_cat  | g_name            | g_price       | g_quantity        | g_top   |            |         |   |          |              |     |   |        |   |   |        |     |    |   |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |
| 1              | 1  | i7                | 250           | 5                 | 0       |            |         |   |          |              |     |   |        |   |   |        |     |    |   |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |
| 2              | 1  | Athlon            | 250           | 10                | 1       |            |         |   |          |              |     |   |        |   |   |        |     |    |   |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |
| 3              | 2  | HP1010            | 300           | 5                 | 1       |            |         |   |          |              |     |   |        |   |   |        |     |    |   |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |
| 4              | 2  | HP2020            | 400           | 7                 | 0       |            |         |   |          |              |     |   |        |   |   |        |     |    |   |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |
| 5              | 2  | Epson-N999        | 500           | 9                 | 1       |            |         |   |          |              |     |   |        |   |   |        |     |    |   |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |
| `ctg_cpu`      | <table border="1"> <thead> <tr> <th>ctg_cpu_id</th><th>ctg_main_link</th><th>ctg_cpu_frequency</th></tr> </thead> <tbody> <tr> <td>1</td><td>1</td><td>3500</td></tr> <tr> <td>2</td><td>2</td><td>3700</td></tr> </tbody> </table>  | ctg_cpu_id        | ctg_main_link | ctg_cpu_frequency | 1       | 1          | 3500    | 2 | 2        | 3700         |     |   |        |   |   |        |     |    |   |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |
| ctg_cpu_id     | ctg_main_link  | ctg_cpu_frequency |               |                   |         |            |         |   |          |              |     |   |        |   |   |        |     |    |   |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |
| 1              | 1  | 3500              |               |                   |         |            |         |   |          |              |     |   |        |   |   |        |     |    |   |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |
| 2              | 2  | 3700              |               |                   |         |            |         |   |          |              |     |   |        |   |   |        |     |    |   |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |
| `ctg_printers` | <table border="1"> <thead> <tr> <th>ctg_prn_id</th><th>ctg_main_link</th><th>ctg_prn_type</th></tr> </thead> <tbody> <tr> <td>1</td><td>3</td><td>laser</td></tr> <tr> <td>2</td><td>4</td><td>laser</td></tr> <tr> <td>3</td><td>5</td><td>inkjet</td></tr> </tbody> </table>   | ctg_prn_id        | ctg_main_link | ctg_prn_type      | 1       | 3          | laser   | 2 | 4        | laser        | 3   | 5 | inkjet |   |   |        |     |    |   |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |
| ctg_prn_id     | ctg_main_link  | ctg_prn_type      |               |                   |         |            |         |   |          |              |     |   |        |   |   |        |     |    |   |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |
| 1              | 3  | laser             |               |                   |         |            |         |   |          |              |     |   |        |   |   |        |     |    |   |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |
| 2              | 4  | laser             |               |                   |         |            |         |   |          |              |     |   |        |   |   |        |     |    |   |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |
| 3              | 5  | inkjet            |               |                   |         |            |         |   |          |              |     |   |        |   |   |        |     |    |   |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |

Показать все товары заданной категории (id категории мы знаем):

| Запрос    | <pre>SET @cat_num = 1; -- Это известный нам заранее id категории. SET @query := concat('SELECT * FROM `goods` JOIN ', (SELECT `c_table` FROM `categories` WHERE `c_id`=@cat_num), ' ON `g_id`='ctg_main_link"); PREPARE final_query FROM @query; EXECUTE final_query;</pre>   |        |         |            |         |            |               |                   |               |                   |   |   |    |     |   |   |   |   |      |   |   |        |     |    |   |   |   |      |
|-----------|---|--------|---------|------------|---------|------------|---------------|-------------------|---------------|-------------------|---|---|----|-----|---|---|---|---|------|---|---|--------|-----|----|---|---|---|------|
| Результат | <table border="1"> <thead> <tr> <th>g_id</th><th>g_cat</th><th>g_name</th><th>g_price</th><th>g_quantity</th><th>g_top</th><th>ctg_cpu_id</th><th>ctg_main_link</th><th>ctg_cpu_frequency</th></tr> </thead> <tbody> <tr> <td>1</td><td>1</td><td>i7</td><td>250</td><td>5</td><td>0</td><td>1</td><td>1</td><td>3500</td></tr> <tr> <td>2</td><td>1</td><td>Athlon</td><td>250</td><td>10</td><td>1</td><td>2</td><td>2</td><td>3700</td></tr> </tbody> </table> | g_id   | g_cat   | g_name     | g_price | g_quantity | g_top         | ctg_cpu_id        | ctg_main_link | ctg_cpu_frequency | 1 | 1 | i7 | 250 | 5 | 0 | 1 | 1 | 3500 | 2 | 1 | Athlon | 250 | 10 | 1 | 2 | 2 | 3700 |
| g_id      | g_cat   | g_name | g_price | g_quantity | g_top   | ctg_cpu_id | ctg_main_link | ctg_cpu_frequency |               |                   |   |   |    |     |   |   |   |   |      |   |   |        |     |    |   |   |   |      |
| 1         | 1   | i7     | 250     | 5          | 0       | 1          | 1             | 3500              |               |                   |   |   |    |     |   |   |   |   |      |   |   |        |     |    |   |   |   |      |
| 2         | 1   | Athlon | 250     | 10         | 1       | 2          | 2             | 3700              |               |                   |   |   |    |     |   |   |   |   |      |   |   |        |     |    |   |   |   |      |

Показать все топовые товары по убыванию цены с их категориями:

| Запрос    | <pre>SELECT * FROM `goods` JOIN `categories` ON `g_cat`='c_id` WHERE `g_top`=1 ORDER BY `g_price` DESC</pre>   |            |         |            |         |            |          |              |        |         |   |   |            |     |   |   |   |          |              |   |   |        |     |   |   |   |          |              |   |   |        |     |    |   |   |     |         |
|-----------|--|------------|---------|------------|---------|------------|----------|--------------|--------|---------|---|---|------------|-----|---|---|---|----------|--------------|---|---|--------|-----|---|---|---|----------|--------------|---|---|--------|-----|----|---|---|-----|---------|
| Результат | <table border="1"> <thead> <tr> <th>g_id</th><th>g_cat</th><th>g_name</th><th>g_price</th><th>g_quantity</th><th>g_top</th><th>c_id</th><th>c_name</th><th>c_table</th></tr> </thead> <tbody> <tr> <td>5</td><td>2</td><td>Epson-N999</td><td>500</td><td>9</td><td>1</td><td>2</td><td>Printers</td><td>ctg_printers</td></tr> <tr> <td>3</td><td>2</td><td>HP1010</td><td>300</td><td>5</td><td>1</td><td>2</td><td>Printers</td><td>ctg_printers</td></tr> <tr> <td>2</td><td>1</td><td>Athlon</td><td>250</td><td>10</td><td>1</td><td>1</td><td>CPU</td><td>ctg_cpu</td></tr> </tbody> </table> | g_id       | g_cat   | g_name     | g_price | g_quantity | g_top    | c_id         | c_name | c_table | 5 | 2 | Epson-N999 | 500 | 9 | 1 | 2 | Printers | ctg_printers | 3 | 2 | HP1010 | 300 | 5 | 1 | 2 | Printers | ctg_printers | 2 | 1 | Athlon | 250 | 10 | 1 | 1 | CPU | ctg_cpu |
| g_id      | g_cat  | g_name     | g_price | g_quantity | g_top   | c_id       | c_name   | c_table      |        |         |   |   |            |     |   |   |   |          |              |   |   |        |     |   |   |   |          |              |   |   |        |     |    |   |   |     |         |
| 5         | 2  | Epson-N999 | 500     | 9          | 1       | 2          | Printers | ctg_printers |        |         |   |   |            |     |   |   |   |          |              |   |   |        |     |   |   |   |          |              |   |   |        |     |    |   |   |     |         |
| 3         | 2  | HP1010     | 300     | 5          | 1       | 2          | Printers | ctg_printers |        |         |   |   |            |     |   |   |   |          |              |   |   |        |     |   |   |   |          |              |   |   |        |     |    |   |   |     |         |
| 2         | 1  | Athlon     | 250     | 10         | 1       | 1          | CPU      | ctg_cpu      |        |         |   |   |            |     |   |   |   |          |              |   |   |        |     |   |   |   |          |              |   |   |        |     |    |   |   |     |         |

Хотите поэкспериментировать? Вот дампы баз!

| БД «Деканат»  | БД «Магазин»   |
|---|--|
| <pre>SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO"; SET time_zone = "+00:00";  /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */; /*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */; /*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */; /*!40101 SET NAMES utf8 */;  -- Database: `joins_2`  -- Table structure for table `awards` --  CREATE TABLE IF NOT EXISTS `awards` (   `a_id` int(11) NOT NULL AUTO_INCREMENT,   `a_mark` int(11) DEFAULT NULL,   `a_name` varchar(255) NOT NULL,   PRIMARY KEY (`a_id`) ) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=6 ;  -- Dumping data for table `awards` --  INSERT INTO `awards` (`a_id`, `a_mark`, `a_name`) VALUES (1, NULL, 'iPhone'), (2, 5, 'Nokia3310'), (3, 4, 'Vodka 0.5'), (4, 9, 'Vodka 1.0'), (5, NULL, 'Dismissal');  -- Table structure for table `facts`  CREATE TABLE IF NOT EXISTS `facts` (   `f_id` int(11) NOT NULL AUTO_INCREMENT,   `sb_id` int(11) NOT NULL,   `tt_id` int(11) NOT NULL,   `st_id` int(11) NOT NULL,   `hours` int(11) DEFAULT NULL,   `type` int(11) DEFAULT NULL,   `f_time` int(11) NOT NULL,   PRIMARY KEY (`f_id`),   UNIQUE KEY `sb_id`(`sb_id`,`tt_id`,`st_id`,`hours`,`type`),   KEY `st_id`(`st_id`),   KEY `tt_id`(`tt_id`),   KEY `type`(`type`),   KEY `f_time`(`f_time`) ) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=12 ;</pre> | <pre>SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO"; SET time_zone = "+00:00";  /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */; /*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */; /*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */; /*!40101 SET NAMES utf8 */;  -- Database: `shop`  -- Table structure for table `categories`  CREATE TABLE IF NOT EXISTS `categories` (   `c_id` int(11) NOT NULL AUTO_INCREMENT,   `c_name` varchar(255) NOT NULL,   `c_table` varchar(255) NOT NULL,   PRIMARY KEY (`c_id`) ) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=3 ;  -- Dumping data for table `categories` --  INSERT INTO `categories` (`c_id`, `c_name`, `c_table`) VALUES (1, 'CPU', 'ctg_cpu'), (2, 'Printers', 'ctg_printers');  -- Table structure for table `ctg_cpu`  CREATE TABLE IF NOT EXISTS `ctg_cpu` (   `ctg_cpu_id` int(11) NOT NULL AUTO_INCREMENT,   `ctg_main_link` int(11) DEFAULT NULL,   `ctg_cpu_frequency` int(11) NOT NULL,   PRIMARY KEY (`ctg_cpu_id`),   UNIQUE KEY `ctg_main_link`(`ctg_main_link`) ) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=4 ;</pre> |

```

-- 
-- Dumping data for table `facts` 
-- 

INSERT INTO `facts` (`f_id`, `sb_id`, `tt_id`, `st_id`, `hours`, `type`, `f_time`) VALUES
(1, 1, 1, 1, 32, 1, 0),
(2, 1, 2, 2, NULL, 1, 0),
(3, 2, 3, 4, 64, 4, 0),
(4, 2, 4, 5, 16, 2, 0),
(5, 3, 1, 1, NULL, 1, 0),
(6, 4, 1, 2, NULL, 1, 0),
(7, 4, 2, 4, NULL, 1, 0),
(8, 4, 4, 5, NULL, 1, 0),
(9, 5, 1, 1, 32, 2, 0),
(10, 5, 5, 2, 128, 1, 0),
(11, 6, 6, 6, 64, 1, 0);

-- 
-- Table structure for table `marks` 
-- 

CREATE TABLE IF NOT EXISTS `marks` (
  `m_id` int(11) NOT NULL AUTO_INCREMENT,
  `m_mark` int(11) NOT NULL,
  `m_date` date NOT NULL,
  `m_fact` int(11) NOT NULL,
  PRIMARY KEY (`m_id`),
  KEY `m_fact` (`m_fact`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=13 ;

-- 
-- Dumping data for table `marks` 
-- 

INSERT INTO `marks` (`m_id`, `m_mark`, `m_date`, `m_fact`) VALUES
(1, 10, '2012-10-09', 1),
(2, 8, '2012-10-10', 1),
(3, 5, '2012-10-24', 2),
(4, 3, '2012-10-23', 2),
(5, 7, '2012-10-09', 3),
(6, 10, '2012-10-17', 3),
(7, 10, '2012-10-08', 3),
(8, 9, '2012-10-16', 4),
(9, 9, '2012-10-17', 4),
(10, 8, '2012-10-02', 4),
(11, 10, '2012-10-01', 11),
(12, 5, '2012-10-08', 11);

-- 
-- Table structure for table `students` 
-- 

CREATE TABLE IF NOT EXISTS `students` (
  `st_id` int(11) NOT NULL AUTO_INCREMENT,
  `st_name` varchar(255) NOT NULL,
  PRIMARY KEY (`st_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=7 ;

-- 
-- Dumping data for table `students` 
-- 

INSERT INTO `students` (`st_id`, `st_name`) VALUES
(1, 'ST_Clever_A'),
(2, 'ST_Clever_B'),
(4, 'ST_Stupid_B'),
(5, 'ST_Average'),
(6, 'TestStudent');

-- 
-- Table structure for table `subjects` 
-- 

CREATE TABLE IF NOT EXISTS `subjects` (
  `sb_id` int(11) NOT NULL AUTO_INCREMENT,
  `sb_name` varchar(255) NOT NULL,
  PRIMARY KEY (`sb_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=7 ;

-- 
-- Dumping data for table `subjects` 
-- 

INSERT INTO `subjects` (`sb_id`, `sb_name`) VALUES
(1, 'SB_OOP'),
(2, 'SB_DB'),
(3, 'SB_OSISP'),
(4, 'SB_SFT'),
(5, 'SB_OhranaTruda'),
(6, 'TestSubject');

-- 
-- Table structure for table `time` 
-- 

```

---

```

-- Dumping data for table `ctg_cpu` 
-- 

INSERT INTO `ctg_cpu` (`ctg_cpu_id`, `ctg_main_link`, `ctg_cpu_frequency`) VALUES
(1, 1, 3500),
(2, 2, 3700);

-- 
-- Table structure for table `ctg_printers` 
-- 

CREATE TABLE IF NOT EXISTS `ctg_printers` (
  `ctg_prn_id` int(11) NOT NULL AUTO_INCREMENT,
  `ctg_main_link` int(11) DEFAULT NULL,
  `ctg_prn_type` enum('laser','inkjet') NOT NULL,
  PRIMARY KEY (`ctg_prn_id`),
  UNIQUE KEY `ctg_main_link` (`ctg_main_link`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=5 ;

-- 
-- Dumping data for table `ctg_printers` 
-- 

INSERT INTO `ctg_printers` (`ctg_prn_id`, `ctg_main_link`, `ctg_prn_type`) VALUES
(1, 3, 'laser'),
(2, 4, 'laser'),
(3, 5, 'inkjet');

-- 
-- Table structure for table `goods` 
-- 

CREATE TABLE IF NOT EXISTS `goods` (
  `g_id` int(11) NOT NULL AUTO_INCREMENT,
  `g_cat` int(11) NOT NULL,
  `g_name` text NOT NULL,
  `g_price` double NOT NULL,
  `g_quantity` int(11) NOT NULL,
  `g_top` bit(1) NOT NULL DEFAULT b'0',
  PRIMARY KEY (`g_id`),
  KEY `g_cat` (`g_cat`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=12 ;

-- 
-- Dumping data for table `goods` 
-- 

INSERT INTO `goods` (`g_id`, `g_cat`, `g_name`, `g_price`, `g_quantity`, `g_top`) VALUES
(1, 1, 'i7', 250, 5, b'0'),
(2, 1, 'Athlon', 250, 10, b'1'),
(3, 2, 'HP1010', 300, 5, b'1'),
(4, 2, 'HP2020', 400, 7, b'0'),
(5, 2, 'Epson-N999', 500, 9, b'1');

-- 
-- Constraints for dumped tables 
-- 

-- 
-- Constraints for table `ctg_cpu` 
-- 

ALTER TABLE `ctg_cpu`
  ADD CONSTRAINT `ctg_cpu_ibfk_1` FOREIGN KEY (`ctg_main_link`) REFERENCES `goods` (`g_id`) ON DELETE CASCADE ON UPDATE CASCADE;

-- 
-- Constraints for table `ctg_printers` 
-- 

ALTER TABLE `ctg_printers`
  ADD CONSTRAINT `ctg_printers_ibfk_1` FOREIGN KEY (`ctg_main_link`) REFERENCES `goods` (`g_id`) ON DELETE CASCADE ON UPDATE CASCADE;

-- 
-- Constraints for table `goods` 
-- 

ALTER TABLE `goods`
  ADD CONSTRAINT `goods_ibfk_1` FOREIGN KEY (`g_cat`) REFERENCES `categories` (`c_id`) ON DELETE CASCADE ON UPDATE CASCADE;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

```

CREATE TABLE IF NOT EXISTS `time` (
  `t_id` int(11) NOT NULL,
  `t_start` date NOT NULL,
  `t_end` date NOT NULL,
  `t_descr` varchar(255) NOT NULL,
  PRIMARY KEY (`t_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- 
-- Dumping data for table `time`
-- 

INSERT INTO `time` (`t_id`, `t_start`, `t_end`, `t_descr`) VALUES
(0, '2012-11-14', '2012-11-24', '');

-- 
-- Table structure for table `tutors`


CREATE TABLE IF NOT EXISTS `tutors` (
  `tt_id` int(11) NOT NULL AUTO_INCREMENT,
  `tt_name` varchar(255) NOT NULL,
  PRIMARY KEY (`tt_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=7 ;

-- 
-- Dumping data for table `tutors`
-- 

INSERT INTO `tutors` (`tt_id`, `tt_name`) VALUES
(1, 'TT_Strict_A'),
(2, 'TT_Strict_B'),
(3, 'TT_Kind_A'),
(4, 'TT_Kind_B'),
(5, 'TT_Spontaneous'),
(6, 'TestTutor');

-- 
-- Table structure for table `types`


CREATE TABLE IF NOT EXISTS `types` (
  `t_id` int(11) NOT NULL AUTO_INCREMENT,
  `t_type` enum('lecture','lab','practice','course','diploma','d_practice','p_practice') NOT NULL,
  PRIMARY KEY (`t_id`),
  UNIQUE KEY `t_type` (`t_type`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=6 ;

-- 
-- Dumping data for table `types`
-- 

INSERT INTO `types` (`t_id`, `t_type`) VALUES
(1, 'lecture'),
(2, 'lab'),
(3, 'practice'),
(4, 'course'),
(5, 'diploma');

-- 
-- Constraints for dumped tables
-- 

-- 
-- Constraints for table `facts`


ALTER TABLE `facts`
  ADD CONSTRAINT `facts_ibfk_5` FOREIGN KEY (`f_time`) REFERENCES `time`(`t_id`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `facts_ibfk_1` FOREIGN KEY (`st_id`) REFERENCES `students`(`st_id`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `facts_ibfk_2` FOREIGN KEY (`tt_id`) REFERENCES `tutors`(`tt_id`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `facts_ibfk_3` FOREIGN KEY (`sb_id`) REFERENCES `subjects`(`sb_id`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `facts_ibfk_4` FOREIGN KEY (`type`) REFERENCES `types`(`t_id`) ON DELETE CASCADE ON UPDATE CASCADE;

-- 
-- Constraints for table `marks`


ALTER TABLE `marks`
  ADD CONSTRAINT `marks_ibfk_1` FOREIGN KEY (`m_fact`) REFERENCES `facts`(`f_id`) ON DELETE CASCADE ON UPDATE CASCADE;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

## 6.5. Подзапросы (SUBQUERIES)

### 6.5.1. Общие сведения о подзапросах

Подзапросы – это «запросы внутри запроса» (вложенные запросы), которые используются как источник данных для:

- получения некоего значения, используемого в операциях сравнения;
- получения набора данных, из которого в дальнейшем производится выборка «основной частью запроса».

Например: удалить из основной таблицы все статьи, копии которых есть в архивной таблице.

```
DELETE FROM `articles_main`
WHERE `a_id` IN (SELECT `a_id` FROM `articles_archive`)
```

Подзапросы могут возвращать:

- одно значение – скалярные подзапросы;
- одну строку – строчные подзапросы;
- одну колонку – столбцовые подзапросы;
- таблицу – табличные подзапросы.

Каждый из этих видов подзапросов может использоваться в своём определённом контексте.

### 6.5.2. Скалярные подзапросы и сравнение с использованием подзапросов

Скалярный подзапрос возвращает одно значение.

Если в результате выполнения подзапроса получается пустой результат, значением подзапроса считается NULL.

Подзапросы не могут использоваться в случае, если синтаксис требует явного указания значений (например, в LIMIT).

Подробности можно изучить здесь:

<http://dev.mysql.com/doc/refman/5.5/en/scalar-subqueries.html>

Пример: показать людей, чей возраст больше среднего возраста.

| Исходная таблица  | Запрос             | Результат          |                   |   |      |    |   |      |    |   |      |    |   |     |    |   |     |    |  |  |                  |                    |                   |   |     |    |
|---|--------------------|--------------------|-------------------|---|------|----|---|------|----|---|------|----|---|-----|----|---|-----|----|--|--|------------------|--------------------|-------------------|---|-----|----|
| <table border="1"> <thead> <tr> <th><b><i>id</i></b></th><th><b><i>name</i></b></th><th><b><i>age</i></b></th></tr> </thead> <tbody> <tr> <td>1</td><td>Jack</td><td>20</td></tr> <tr> <td>2</td><td>John</td><td>30</td></tr> <tr> <td>3</td><td>Alex</td><td>35</td></tr> <tr> <td>4</td><td>Bob</td><td>40</td></tr> <tr> <td>5</td><td>Joe</td><td>90</td></tr> </tbody> </table> | <b><i>id</i></b>   | <b><i>name</i></b> | <b><i>age</i></b> | 1 | Jack | 20 | 2 | John | 30 | 3 | Alex | 35 | 4 | Bob | 40 | 5 | Joe | 90 | <pre>SELECT * FROM `people` WHERE `age` &gt; (SELECT AVG(`age`) FROM `people`)</pre> | <table border="1"> <thead> <tr> <th><b><i>id</i></b></th><th><b><i>name</i></b></th><th><b><i>age</i></b></th></tr> </thead> <tbody> <tr> <td>5</td><td>Joe</td><td>90</td></tr> </tbody> </table> | <b><i>id</i></b> | <b><i>name</i></b> | <b><i>age</i></b> | 5 | Joe | 90 |
| <b><i>id</i></b>  | <b><i>name</i></b> | <b><i>age</i></b>  |                   |   |      |    |   |      |    |   |      |    |   |     |    |   |     |    |  |  |                  |                    |                   |   |     |    |
| 1   | Jack               | 20                 |                   |   |      |    |   |      |    |   |      |    |   |     |    |   |     |    |  |  |                  |                    |                   |   |     |    |
| 2   | John               | 30                 |                   |   |      |    |   |      |    |   |      |    |   |     |    |   |     |    |  |  |                  |                    |                   |   |     |    |
| 3   | Alex               | 35                 |                   |   |      |    |   |      |    |   |      |    |   |     |    |   |     |    |  |  |                  |                    |                   |   |     |    |
| 4   | Bob                | 40                 |                   |   |      |    |   |      |    |   |      |    |   |     |    |   |     |    |  |  |                  |                    |                   |   |     |    |
| 5   | Joe                | 90                 |                   |   |      |    |   |      |    |   |      |    |   |     |    |   |     |    |  |  |                  |                    |                   |   |     |    |
| <b><i>id</i></b>  | <b><i>name</i></b> | <b><i>age</i></b>  |                   |   |      |    |   |      |    |   |      |    |   |     |    |   |     |    |  |  |                  |                    |                   |   |     |    |
| 5   | Joe                | 90                 |                   |   |      |    |   |      |    |   |      |    |   |     |    |   |     |    |  |  |                  |                    |                   |   |     |    |

Общий принцип использования подзапросов в операциях сравнения может быть выражен так:

ОПЕРАНД операция\_сравнения (ПОДЗАПРОС)

Операциями сравнения могут быть:

= > < >= <= <> != <=> LIKE

**Особенность MySQL.**

Подробности можно изучить здесь:

<http://dev.mysql.com/doc/refman/5.5/en/comparisons-using-subqueries.html>

Как правило, сравнение с использованием запросов используется в случае, если запрос невозможно (или вычислительно сложно) реализовать с помощью JOIN.

Одним из таких примеров может быть уже рассмотренный поиск людей с возрастом больше среднего:

```
SELECT * FROM `people`
WHERE `age` > (SELECT AVG(`age`) FROM `people`)
```

Ещё один пример – поиск записей, результат выборки которых с некоторым полем, встречается заданное количество раз.

Например: найти всех однофамильцев, которых в таблице по трое.

| Исходная таблица   |             |                | Запрос         | Результат |     |       |   |      |       |   |      |       |   |      |       |   |       |       |   |        |       |   |      |         |   |         |        |   |      |          |    |      |          |  |  |  |             |     |      |      |      |       |        |
|--|-------------|----------------|----------------|-----------|-----|-------|---|------|-------|---|------|-------|---|------|-------|---|-------|-------|---|--------|-------|---|------|---------|---|---------|--------|---|------|----------|----|------|----------|--|--|--|-------------|-----|------|------|------|-------|--------|
| <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>surname</b></th></tr> </thead> <tbody> <tr><td>1</td><td>Joe</td><td>Black</td></tr> <tr><td>2</td><td>John</td><td>Black</td></tr> <tr><td>3</td><td>Alex</td><td>Black</td></tr> <tr><td>4</td><td>John</td><td>Smith</td></tr> <tr><td>5</td><td>Helen</td><td>Smith</td></tr> <tr><td>6</td><td>Jakkie</td><td>Smith</td></tr> <tr><td>7</td><td>Ivan</td><td>Susanin</td></tr> <tr><td>8</td><td>Vasiliy</td><td>Pupkin</td></tr> <tr><td>9</td><td>Ilya</td><td>Morumets</td></tr> <tr><td>10</td><td>Zmey</td><td>Gorynych</td></tr> </tbody> </table> | <b>id</b>   | <b>name</b>    | <b>surname</b> | 1         | Joe | Black | 2 | John | Black | 3 | Alex | Black | 4 | John | Smith | 5 | Helen | Smith | 6 | Jakkie | Smith | 7 | Ivan | Susanin | 8 | Vasiliy | Pupkin | 9 | Ilya | Morumets | 10 | Zmey | Gorynych |  | <pre>SELECT `name` FROM `people2` AS `OuterTable` WHERE (SELECT COUNT(*) FROM `people2` AS `InnerTable` WHERE `OuterTable`.`surname` = `InnerTable`.`Surname`) = 3</pre> | <table border="1"> <thead> <tr> <th><b>name</b></th></tr> </thead> <tbody> <tr><td>Joe</td></tr> <tr><td>John</td></tr> <tr><td>Alex</td></tr> <tr><td>John</td></tr> <tr><td>Helen</td></tr> <tr><td>Jakkie</td></tr> </tbody> </table> | <b>name</b> | Joe | John | Alex | John | Helen | Jakkie |
| <b>id</b>  | <b>name</b> | <b>surname</b> |                |           |     |       |   |      |       |   |      |       |   |      |       |   |       |       |   |        |       |   |      |         |   |         |        |   |      |          |    |      |          |  |  |  |             |     |      |      |      |       |        |
| 1  | Joe         | Black          |                |           |     |       |   |      |       |   |      |       |   |      |       |   |       |       |   |        |       |   |      |         |   |         |        |   |      |          |    |      |          |  |  |  |             |     |      |      |      |       |        |
| 2  | John        | Black          |                |           |     |       |   |      |       |   |      |       |   |      |       |   |       |       |   |        |       |   |      |         |   |         |        |   |      |          |    |      |          |  |  |  |             |     |      |      |      |       |        |
| 3  | Alex        | Black          |                |           |     |       |   |      |       |   |      |       |   |      |       |   |       |       |   |        |       |   |      |         |   |         |        |   |      |          |    |      |          |  |  |  |             |     |      |      |      |       |        |
| 4  | John        | Smith          |                |           |     |       |   |      |       |   |      |       |   |      |       |   |       |       |   |        |       |   |      |         |   |         |        |   |      |          |    |      |          |  |  |  |             |     |      |      |      |       |        |
| 5  | Helen       | Smith          |                |           |     |       |   |      |       |   |      |       |   |      |       |   |       |       |   |        |       |   |      |         |   |         |        |   |      |          |    |      |          |  |  |  |             |     |      |      |      |       |        |
| 6  | Jakkie      | Smith          |                |           |     |       |   |      |       |   |      |       |   |      |       |   |       |       |   |        |       |   |      |         |   |         |        |   |      |          |    |      |          |  |  |  |             |     |      |      |      |       |        |
| 7  | Ivan        | Susanin        |                |           |     |       |   |      |       |   |      |       |   |      |       |   |       |       |   |        |       |   |      |         |   |         |        |   |      |          |    |      |          |  |  |  |             |     |      |      |      |       |        |
| 8  | Vasiliy     | Pupkin         |                |           |     |       |   |      |       |   |      |       |   |      |       |   |       |       |   |        |       |   |      |         |   |         |        |   |      |          |    |      |          |  |  |  |             |     |      |      |      |       |        |
| 9  | Ilya        | Morumets       |                |           |     |       |   |      |       |   |      |       |   |      |       |   |       |       |   |        |       |   |      |         |   |         |        |   |      |          |    |      |          |  |  |  |             |     |      |      |      |       |        |
| 10   | Zmey        | Gorynych       |                |           |     |       |   |      |       |   |      |       |   |      |       |   |       |       |   |        |       |   |      |         |   |         |        |   |      |          |    |      |          |  |  |  |             |     |      |      |      |       |        |
| <b>name</b>  |             |                |                |           |     |       |   |      |       |   |      |       |   |      |       |   |       |       |   |        |       |   |      |         |   |         |        |   |      |          |    |      |          |  |  |  |             |     |      |      |      |       |        |
| Joe  |             |                |                |           |     |       |   |      |       |   |      |       |   |      |       |   |       |       |   |        |       |   |      |         |   |         |        |   |      |          |    |      |          |  |  |  |             |     |      |      |      |       |        |
| John   |             |                |                |           |     |       |   |      |       |   |      |       |   |      |       |   |       |       |   |        |       |   |      |         |   |         |        |   |      |          |    |      |          |  |  |  |             |     |      |      |      |       |        |
| Alex   |             |                |                |           |     |       |   |      |       |   |      |       |   |      |       |   |       |       |   |        |       |   |      |         |   |         |        |   |      |          |    |      |          |  |  |  |             |     |      |      |      |       |        |
| John   |             |                |                |           |     |       |   |      |       |   |      |       |   |      |       |   |       |       |   |        |       |   |      |         |   |         |        |   |      |          |    |      |          |  |  |  |             |     |      |      |      |       |        |
| Helen  |             |                |                |           |     |       |   |      |       |   |      |       |   |      |       |   |       |       |   |        |       |   |      |         |   |         |        |   |      |          |    |      |          |  |  |  |             |     |      |      |      |       |        |
| Jakkie   |             |                |                |           |     |       |   |      |       |   |      |       |   |      |       |   |       |       |   |        |       |   |      |         |   |         |        |   |      |          |    |      |          |  |  |  |             |     |      |      |      |       |        |

### 6.5.3. Подзапросы с ключевыми словами ANY, IN, SOME, ALL

Для понимания того, как работают подзапросы с этими ключевыми словами, нужно вспомнить логику.

Проще всего – со словом IN. Оно определяет вхождение (IN) или не вхождение (NOT IN) элемента в некий набор.

|   |        |                     |     |
|---|--------|---------------------|-----|
| 5 | IN     | 10, 2, 5, 45, 2, 11 | OK  |
| 7 | IN     | 10, 2, 5, 45, 2, 11 | NO! |
| 5 | NOT IN | 10, 2, 5, 45, 2, 11 | NO! |
| 7 | NOT IN | 10, 2, 5, 45, 2, 11 | OK  |

Теперь рассмотрим ANY (синоним SOME):

|   |        |                      |     |
|---|--------|----------------------|-----|
| 5 | < ANY  | 10, 2, 5, 45, 2, 11  | OK  |
| 5 | > ANY  | 10, 2, 5, 45, 2, 11  | OK  |
| 5 | = ANY  | 10, 2, 5, 45, 2, 11  | OK  |
| 5 | != ANY | 10, 2, 5, 45, 2, 11  | OK  |
| 5 | < ANY  | 0, 1, 2, 1, -5, 3, 4 | NO! |
| 5 | > ANY  | 0, 1, 2, 1, -5, 3, 4 | OK  |
| 5 | = ANY  | 0, 1, 2, 1, -5, 3, 4 | NO! |

И, наконец, рассмотрим ALL:

|   |        |                        |     |
|---|--------|------------------------|-----|
| 5 | < ALL  | 10, 2, 5, 45, 2, 11    | NO! |
| 5 | > ALL  | 10, 2, 5, 45, 2, 11    | NO! |
| 5 | = ALL  | 10, 2, 5, 45, 2, 11    | NO! |
| 5 | != ALL | 10, 2, 5, 45, 2, 11    | NO! |
| 5 | < ALL  | 0, 1, 2, 1, -5, 3, 4   | NO! |
| 5 | > ALL  | 0, 1, 2, 1, -5, 3, 4   | OK  |
| 5 | > ALL  | 0, 1, 2, 1, -5, 3, 999 | NO! |

Пример 1 (очень жизненный, часто бывают ошибки): удалить статьи, id которых нам известны.

Варианты решений:

[ПЛОХО!] Цикл с запросами вида:

```
DELETE ... WHERE `id` = {id}
```

[ПЛОХО!] Запрос вида:

```
DELETE ... WHERE `id`={id1} OR `id`={id2} OR ... OR `id`={id999}
```

[ХОРОШО!] Запрос вида:

```
DELETE ... WHERE `id` IN (10, 14, 87, 174)
```

[ХОРОШО!] Запрос вида:

```
DELETE ... WHERE `id` IN (SELECT `id` FROM `victims`)
```

Пример 2: показать людей моложе ЛЮБОГО из указанных возрастов.

В ANY нельзя передать список значений.

Такой вариант НЕ работает:

```
SELECT `name`, `age` FROM `people` WHERE `age` < ANY (35, 45, 55)
```

Работает вот такой:

| Исходные таблицы      |                   |                     | Запрос  | Результат  |                   |                  |      |    |      |    |      |    |     |    |
|-----------------------|-------------------|---------------------|---|--|-------------------|------------------|------|----|------|----|------|----|-----|----|
|                       |                   |                     |   |  |                   |                  |      |    |      |    |      |    |     |    |
| <code>`people`</code> |                   | <code>`ages`</code> | <code>SELECT `name`, `age` FROM `people` WHERE `age` &lt; ANY (SELECT `age` FROM `ages`)</code> |  |                   |                  |      |    |      |    |      |    |     |    |
| <code>id</code>       | <code>name</code> | <code>age</code>    |   |  |                   |                  |      |    |      |    |      |    |     |    |
| 1                     | Jack              | 20                  |   | <table border="1"> <thead> <tr> <th><code>name</code></th><th><code>age</code></th></tr> </thead> <tbody> <tr> <td>Jack</td><td>20</td></tr> <tr> <td>John</td><td>30</td></tr> <tr> <td>Alex</td><td>35</td></tr> <tr> <td>Bob</td><td>40</td></tr> </tbody> </table> | <code>name</code> | <code>age</code> | Jack | 20 | John | 30 | Alex | 35 | Bob | 40 |
| <code>name</code>     | <code>age</code>  |                     |   |  |                   |                  |      |    |      |    |      |    |     |    |
| Jack                  | 20                |                     |   |  |                   |                  |      |    |      |    |      |    |     |    |
| John                  | 30                |                     |   |  |                   |                  |      |    |      |    |      |    |     |    |
| Alex                  | 35                |                     |   |  |                   |                  |      |    |      |    |      |    |     |    |
| Bob                   | 40                |                     |   |  |                   |                  |      |    |      |    |      |    |     |    |
| 2                     | John              | 30                  |   |  |                   |                  |      |    |      |    |      |    |     |    |
| 3                     | Alex              | 35                  |   |  |                   |                  |      |    |      |    |      |    |     |    |
| 4                     | Bob               | 40                  |   |  |                   |                  |      |    |      |    |      |    |     |    |
| 5                     | Joe               | 90                  |   |  |                   |                  |      |    |      |    |      |    |     |    |

Эту задачу можно решить и другим способом, с использованием функции MAX().

Так же нельзя.

Такой запрос тоже не работает:

```
SELECT `name`, `age` FROM `people` WHERE `age` < MAX (35, 45, 55)
```

А такой вернёт результат, аналогичный запросу с ANY:

```
SELECT `name`, `age` FROM `people`
WHERE `age` < (SELECT MAX(`age`) FROM `ages`)
```

Пример 3: показать, сколько в таблице людей, моложе ВСЕХ указанных возрастов.

| Исходные таблицы      |  |   | Запрос  | Результат   |      |     |      |    |      |    |
|-----------------------|--|---|---|---|------|-----|------|----|------|----|
|                       |  |   |   |   |      |     |      |    |      |    |
| <code>`people`</code> | <code>id</code> <code>name</code> <code>age</code> | <code>`ages`</code><br><code>age</code> | <pre>SELECT `name`, `age` FROM `people` WHERE `age` &lt; ALL (SELECT `age` FROM `ages`)</pre> | <table border="1"> <thead> <tr> <th>name</th> <th>age</th> </tr> </thead> <tbody> <tr> <td>Jack</td> <td>20</td> </tr> <tr> <td>John</td> <td>30</td> </tr> </tbody> </table> | name | age | Jack | 20 | John | 30 |
| name                  | age  |   |   |   |      |     |      |    |      |    |
| Jack                  | 20   |   |   |   |      |     |      |    |      |    |
| John                  | 30   |   |   |   |      |     |      |    |      |    |
| 1                     | Jack   | 20                                      | 35  |   |      |     |      |    |      |    |
| 2                     | John   | 30                                      | 45  |   |      |     |      |    |      |    |
| 3                     | Alex   | 35                                      | 55  |   |      |     |      |    |      |    |
| 4                     | Bob  | 40                                      |   |   |      |     |      |    |      |    |
| 5                     | Joe  | 90                                      |   |   |      |     |      |    |      |    |

Второй вариант решения этой задачи выглядит так:

```
SELECT `name`, `age` FROM `people`
WHERE `age` < (SELECT MIN(`age`) FROM `ages`)
```

**Важный момент!** При использовании ALL надо помнить, что результат будет TRUE, если подзапрос вернёт пустое значение. При этом агрегирующие функции (MIN, MAX, etc) на пустом множестве возвращают NULL.

| Исходные таблицы      |  |   | Запросы  | Результаты   |      |     |      |    |      |    |      |    |     |    |     |    |
|-----------------------|--|---|--|--|------|-----|------|----|------|----|------|----|-----|----|-----|----|
|                       |  |   |  |  |      |     |      |    |      |    |      |    |     |    |     |    |
| <code>`people`</code> | <code>id</code> <code>name</code> <code>age</code> | <code>`ages`</code><br><code>age</code> | <pre>SELECT `name`, `age` FROM `people` WHERE `age` &gt; ALL (SELECT `age` FROM `ages`<br/>WHERE `age` = 999)</pre><br><pre>SELECT `name`, `age` FROM `people` WHERE `age` &lt; ALL (SELECT `age` FROM `ages`<br/>WHERE `age` = 999)</pre><br><pre>SELECT `name`, `age` FROM `people` WHERE `age` &lt; (SELECT MAX(`age`) FROM `ages`<br/>WHERE `age` = 999)</pre> | <table border="1"> <thead> <tr> <th>name</th> <th>age</th> </tr> </thead> <tbody> <tr> <td>Jack</td> <td>20</td> </tr> <tr> <td>John</td> <td>30</td> </tr> <tr> <td>Alex</td> <td>35</td> </tr> <tr> <td>Bob</td> <td>40</td> </tr> <tr> <td>Joe</td> <td>90</td> </tr> </tbody> </table> | name | age | Jack | 20 | John | 30 | Alex | 35 | Bob | 40 | Joe | 90 |
| name                  | age  |   |  |  |      |     |      |    |      |    |      |    |     |    |     |    |
| Jack                  | 20   |   |  |  |      |     |      |    |      |    |      |    |     |    |     |    |
| John                  | 30   |   |  |  |      |     |      |    |      |    |      |    |     |    |     |    |
| Alex                  | 35   |   |  |  |      |     |      |    |      |    |      |    |     |    |     |    |
| Bob                   | 40   |   |  |  |      |     |      |    |      |    |      |    |     |    |     |    |
| Joe                   | 90   |   |  |  |      |     |      |    |      |    |      |    |     |    |     |    |
| 1                     | Jack   | 20                                      | 35   |  |      |     |      |    |      |    |      |    |     |    |     |    |
| 2                     | John   | 30                                      | 45   |  |      |     |      |    |      |    |      |    |     |    |     |    |
| 3                     | Alex   | 35                                      | 55   |  |      |     |      |    |      |    |      |    |     |    |     |    |
| 4                     | Bob  | 40                                      |  |  |      |     |      |    |      |    |      |    |     |    |     |    |
| 5                     | Joe  | 90                                      |  |  |      |     |      |    |      |    |      |    |     |    |     |    |

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0005 sec)

### 6.5.4. Подзапросы, возвращающие ряды

Подзапросы, возвращающие ряды, позволяют выполнять сравнение сразу по нескольким полям.

Это эквивалентно выражению вида:

```
`field1a` = `field1b` AND `field2a` = `field2b`
```

Некоторые дополнительные подробности – здесь:

<http://dev.mysql.com/doc/refman/5.5/en/row-subqueries.html>

Пример 1: показать всех соискателей, находящихся в списке рекомендованных.

| Исходные таблицы   |             |                | Запрос       | Результат |  |           |             |                |   |      |       |   |     |       |   |         |        |   |          |         |   |  |           |             |                |   |     |       |
|--|-------------|----------------|--------------|-----------|--|-----------|-------------|----------------|---|------|-------|---|-----|-------|---|---------|--------|---|----------|---------|---|--|-----------|-------------|----------------|---|-----|-------|
| <table border="1"> <thead> <tr> <th colspan="3">`candidates`</th></tr> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>surname</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>John</td><td>Smith</td></tr> <tr> <td>2</td><td>Joe</td><td>Black</td></tr> <tr> <td>3</td><td>Vasiliy</td><td>Pupkin</td></tr> <tr> <td>4</td><td>Vasilisa</td><td>Pupkina</td></tr> </tbody> </table> |             |                | `candidates` |           |  | <b>id</b> | <b>name</b> | <b>surname</b> | 1 | John | Smith | 2 | Joe | Black | 3 | Vasiliy | Pupkin | 4 | Vasilisa | Pupkina | <pre>SELECT * FROM `candidates` WHERE (`name`, `surname`) = (SELECT `name`, `surname` FROM `recommended`)</pre> | <table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>surname</b></th></tr> </thead> <tbody> <tr> <td>2</td><td>Joe</td><td>Black</td></tr> </tbody> </table> | <b>id</b> | <b>name</b> | <b>surname</b> | 2 | Joe | Black |
| `candidates`   |             |                |              |           |  |           |             |                |   |      |       |   |     |       |   |         |        |   |          |         |   |  |           |             |                |   |     |       |
| <b>id</b>  | <b>name</b> | <b>surname</b> |              |           |  |           |             |                |   |      |       |   |     |       |   |         |        |   |          |         |   |  |           |             |                |   |     |       |
| 1  | John        | Smith          |              |           |  |           |             |                |   |      |       |   |     |       |   |         |        |   |          |         |   |  |           |             |                |   |     |       |
| 2  | Joe         | Black          |              |           |  |           |             |                |   |      |       |   |     |       |   |         |        |   |          |         |   |  |           |             |                |   |     |       |
| 3  | Vasiliy     | Pupkin         |              |           |  |           |             |                |   |      |       |   |     |       |   |         |        |   |          |         |   |  |           |             |                |   |     |       |
| 4  | Vasilisa    | Pupkina        |              |           |  |           |             |                |   |      |       |   |     |       |   |         |        |   |          |         |   |  |           |             |                |   |     |       |
| <b>id</b>  | <b>name</b> | <b>surname</b> |              |           |  |           |             |                |   |      |       |   |     |       |   |         |        |   |          |         |   |  |           |             |                |   |     |       |
| 2  | Joe         | Black          |              |           |  |           |             |                |   |      |       |   |     |       |   |         |        |   |          |         |   |  |           |             |                |   |     |       |
|  |             |                |              |           |  |           |             |                |   |      |       |   |     |       |   |         |        |   |          |         |   |  |           |             |                |   |     |       |

Пример 2 (редкий случай, но пример интересный): показать все строки первой таблицы, которые по данному набору полей совпадают со строками другой таблицы. Пример таблиц и результатов не приводим в силу их самоочевидности.

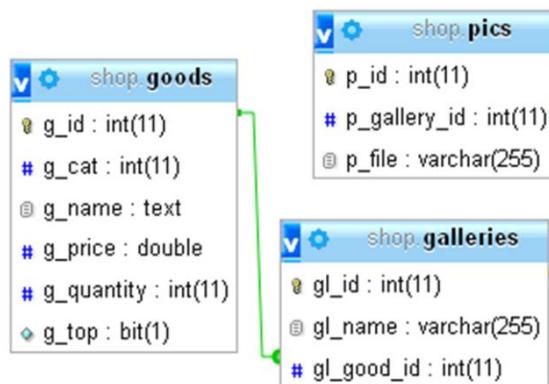
```
SELECT `col1`, `col2`, `col3` FROM `FirstTable`
WHERE (`col1`, `col2`, `col3`) IN
(SELECT `col1`, `col2`, `col3` FROM `SecondTable`)
```

### 6.5.5. Подзапросы с ключевым словом [NOT] EXISTS

Такие подзапросы используются для различения всего лишь двух вариантов:

- подзапрос вернул пустой (`EXISTS == false`) результат или
- подзапрос вернул непустой (`EXISTS == true`) результат..

Рассмотрим на простом примере. Пусть фрагмент базы данных интернет-магазина имеет следующий вид:



Необходимо показать все товары, в галереях которых есть картинки.

| Исходные таблицы | `goods`   | `galleries` | `pics`  |            |         |            |         |            |       |    |     |        |     |    |   |        |     |        |     |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |   |       |         |            |   |             |   |   |             |   |   |             |   |   |             |   |   |      |              |        |   |  |          |   |  |          |
|------------------|---|-------------|---------|------------|---------|------------|---------|------------|-------|----|-----|--------|-----|----|---|--------|-----|--------|-----|---|---|--------|-----|---|---|---|---|--------|-----|---|---|---|---|------------|-----|---|---|---|-------|---------|------------|---|-------------|---|---|-------------|---|---|-------------|---|---|-------------|---|---|------|--------------|--------|---|--|----------|---|--|----------|
|                  | <table border="1"> <thead> <tr> <th>g_id</th><th>g_cat</th><th>g_name</th><th>g_price</th><th>g_quantity</th><th>g_top</th></tr> </thead> <tbody> <tr><td>1</td><td>1</td><td>i7</td><td>250</td><td>5</td><td>0</td></tr> <tr><td>2</td><td>1</td><td>Athlon</td><td>250</td><td>10</td><td>1</td></tr> <tr><td>3</td><td>2</td><td>HP1010</td><td>300</td><td>5</td><td>1</td></tr> <tr><td>4</td><td>2</td><td>HP2020</td><td>400</td><td>7</td><td>0</td></tr> <tr><td>5</td><td>2</td><td>Epson-N999</td><td>500</td><td>9</td><td>1</td></tr> </tbody> </table> | g_id        | g_cat   | g_name     | g_price | g_quantity | g_top   | 1          | 1     | i7 | 250 | 5      | 0   | 2  | 1 | Athlon | 250 | 10     | 1   | 3 | 2 | HP1010 | 300 | 5 | 1 | 4 | 2 | HP2020 | 400 | 7 | 0 | 5 | 2 | Epson-N999 | 500 | 9 | 1 | <table border="1"> <thead> <tr> <th>gl_id</th><th>gl_name</th><th>gl_good_id</th></tr> </thead> <tbody> <tr><td>1</td><td>Gallery-1-1</td><td>1</td></tr> <tr><td>2</td><td>Gallery-2-1</td><td>2</td></tr> <tr><td>3</td><td>Gallery-2-2</td><td>2</td></tr> <tr><td>4</td><td>Gallery-4-1</td><td>4</td></tr> </tbody> </table> | gl_id | gl_name | gl_good_id | 1 | Gallery-1-1 | 1 | 2 | Gallery-2-1 | 2 | 3 | Gallery-2-2 | 2 | 4 | Gallery-4-1 | 4 | <table border="1"> <thead> <tr> <th>p_id</th><th>p_gallery_id</th><th>p_file</th></tr> </thead> <tbody> <tr><td>1</td><td></td><td>pic1.jpg</td></tr> <tr><td>2</td><td></td><td>pic2.jpg</td></tr> </tbody> </table> | p_id | p_gallery_id | p_file | 1 |  | pic1.jpg | 2 |  | pic2.jpg |
| g_id             | g_cat   | g_name      | g_price | g_quantity | g_top   |            |         |            |       |    |     |        |     |    |   |        |     |        |     |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |   |       |         |            |   |             |   |   |             |   |   |             |   |   |             |   |   |      |              |        |   |  |          |   |  |          |
| 1                | 1   | i7          | 250     | 5          | 0       |            |         |            |       |    |     |        |     |    |   |        |     |        |     |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |   |       |         |            |   |             |   |   |             |   |   |             |   |   |             |   |   |      |              |        |   |  |          |   |  |          |
| 2                | 1   | Athlon      | 250     | 10         | 1       |            |         |            |       |    |     |        |     |    |   |        |     |        |     |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |   |       |         |            |   |             |   |   |             |   |   |             |   |   |             |   |   |      |              |        |   |  |          |   |  |          |
| 3                | 2   | HP1010      | 300     | 5          | 1       |            |         |            |       |    |     |        |     |    |   |        |     |        |     |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |   |       |         |            |   |             |   |   |             |   |   |             |   |   |             |   |   |      |              |        |   |  |          |   |  |          |
| 4                | 2   | HP2020      | 400     | 7          | 0       |            |         |            |       |    |     |        |     |    |   |        |     |        |     |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |   |       |         |            |   |             |   |   |             |   |   |             |   |   |             |   |   |      |              |        |   |  |          |   |  |          |
| 5                | 2   | Epson-N999  | 500     | 9          | 1       |            |         |            |       |    |     |        |     |    |   |        |     |        |     |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |   |       |         |            |   |             |   |   |             |   |   |             |   |   |             |   |   |      |              |        |   |  |          |   |  |          |
| gl_id            | gl_name   | gl_good_id  |         |            |         |            |         |            |       |    |     |        |     |    |   |        |     |        |     |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |   |       |         |            |   |             |   |   |             |   |   |             |   |   |             |   |   |      |              |        |   |  |          |   |  |          |
| 1                | Gallery-1-1   | 1           |         |            |         |            |         |            |       |    |     |        |     |    |   |        |     |        |     |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |   |       |         |            |   |             |   |   |             |   |   |             |   |   |             |   |   |      |              |        |   |  |          |   |  |          |
| 2                | Gallery-2-1   | 2           |         |            |         |            |         |            |       |    |     |        |     |    |   |        |     |        |     |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |   |       |         |            |   |             |   |   |             |   |   |             |   |   |             |   |   |      |              |        |   |  |          |   |  |          |
| 3                | Gallery-2-2   | 2           |         |            |         |            |         |            |       |    |     |        |     |    |   |        |     |        |     |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |   |       |         |            |   |             |   |   |             |   |   |             |   |   |             |   |   |      |              |        |   |  |          |   |  |          |
| 4                | Gallery-4-1   | 4           |         |            |         |            |         |            |       |    |     |        |     |    |   |        |     |        |     |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |   |       |         |            |   |             |   |   |             |   |   |             |   |   |             |   |   |      |              |        |   |  |          |   |  |          |
| p_id             | p_gallery_id  | p_file      |         |            |         |            |         |            |       |    |     |        |     |    |   |        |     |        |     |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |   |       |         |            |   |             |   |   |             |   |   |             |   |   |             |   |   |      |              |        |   |  |          |   |  |          |
| 1                |   | pic1.jpg    |         |            |         |            |         |            |       |    |     |        |     |    |   |        |     |        |     |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |   |       |         |            |   |             |   |   |             |   |   |             |   |   |             |   |   |      |              |        |   |  |          |   |  |          |
| 2                |   | pic2.jpg    |         |            |         |            |         |            |       |    |     |        |     |    |   |        |     |        |     |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |   |       |         |            |   |             |   |   |             |   |   |             |   |   |             |   |   |      |              |        |   |  |          |   |  |          |
| Запрос           | <pre>SELECT * FROM `goods` WHERE EXISTS     (SELECT `galleries`.`gl_good_id` FROM `galleries` JOIN `pics`     ON `gl_id` = `p_gallery_id` HAVING `galleries`.`gl_good_id` = `g_id`)</pre>   |             |         |            |         |            |         |            |       |    |     |        |     |    |   |        |     |        |     |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |   |       |         |            |   |             |   |   |             |   |   |             |   |   |             |   |   |      |              |        |   |  |          |   |  |          |
| Результат        | <table border="1"> <thead> <tr> <th>g_id</th><th>g_cat</th><th>g_name</th><th>g_price</th><th>g_quantity</th><th>g_top</th></tr> </thead> <tbody> <tr> <td>2</td><td>1</td><td>Athlon</td><td>250</td><td>10</td><td>1</td></tr> <tr> <td>4</td><td>2</td><td>HP2020</td><td>400</td><td>7</td><td>0</td></tr> </tbody> </table>  |             |         | g_id       | g_cat   | g_name     | g_price | g_quantity | g_top | 2  | 1   | Athlon | 250 | 10 | 1 | 4      | 2   | HP2020 | 400 | 7 | 0 |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |   |       |         |            |   |             |   |   |             |   |   |             |   |   |             |   |   |      |              |        |   |  |          |   |  |          |
| g_id             | g_cat   | g_name      | g_price | g_quantity | g_top   |            |         |            |       |    |     |        |     |    |   |        |     |        |     |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |   |       |         |            |   |             |   |   |             |   |   |             |   |   |             |   |   |      |              |        |   |  |          |   |  |          |
| 2                | 1   | Athlon      | 250     | 10         | 1       |            |         |            |       |    |     |        |     |    |   |        |     |        |     |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |   |       |         |            |   |             |   |   |             |   |   |             |   |   |             |   |   |      |              |        |   |  |          |   |  |          |
| 4                | 2   | HP2020      | 400     | 7          | 0       |            |         |            |       |    |     |        |     |    |   |        |     |        |     |   |   |        |     |   |   |   |   |        |     |   |   |   |   |            |     |   |   |   |       |         |            |   |             |   |   |             |   |   |             |   |   |             |   |   |      |              |        |   |  |          |   |  |          |

### 6.5.6. Взаимосвязанные запросы и подзапросы

Запрос и подзапрос считаются взаимосвязанными, если подзапрос обращается к таблице (или таблицам), которая используется во внешнем запросе.

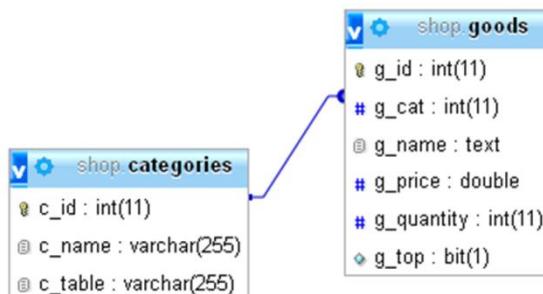
Иллюстрацию такой ситуации мы только что рассматривали в примере про показ товаров, в галереях которых есть картинки.

```
SELECT * FROM `goods` where EXISTS
(SELECT `galleries`.`gl_good_id` FROM `galleries` JOIN `pics` ON
`galleries`.`gl_id` = `pics`.`p_gallery_id` HAVING `galleries`.`gl_good_id` = `goods`.`g_id`)
```

### 6.5.7. Подзапросы как источник данных

В некоторых случаях необходимый результат нельзя получить «за один раз», т.е. необходимо подготовить промежуточный результат, а уже на его основе – конечный. В таких случаях промежуточный результат обычно генерируется подзапросами.

Рассмотрим пример. Необходимо выяснить «среднюю стоимость категорий товаров» (где «стоимость категории» определяется как сумма стоимости всех её товаров). Схема базы данных имеет вид:



| Исходные таблицы |       |            |         |            |       |              | `categories` |              |
|------------------|-------|------------|---------|------------|-------|--------------|--------------|--------------|
| `goods`          |       |            |         |            |       | `categories` |              |              |
| g_id             | g_cat | g_name     | g_price | g_quantity | g_top | c_id         | c_name       | c_table      |
| 1                | 1     | i7         | 250     | 5          | 0     | 1            | CPU          | ctg_cpu      |
| 2                | 1     | Athlon     | 250     | 10         | 1     | 2            | Printers     | ctg_printers |
| 3                | 2     | HP1010     | 300     | 5          | 1     |              |              |              |
| 4                | 2     | HP2020     | 400     | 7          | 0     |              |              |              |
| 5                | 2     | Epson-N999 | 500     | 9          | 1     |              |              |              |

| Запрос   | Результат  |
|--|--|
| <pre>SELECT AVG(SUM(`g_price`)) FROM `goods`<br/>GROUP BY `g_cat`</pre>  | <span style="color: red;">#1111 - Invalid use of group function</span> |
| <pre>SELECT AVG(`inner_sum`) FROM<br/>(SELECT SUM(`g_price`) AS `inner_sum` FROM<br/>`goods` GROUP BY `g_cat`) AS `inner_result`</pre> | Avg(inner_sum)<br>850  |

Для наглядности посмотрим, что возвращает подзапрос:

| Подзапрос   | Результат                |
|---|--------------------------|
| <pre>SELECT SUM(`g_price`) AS `inner_sum` FROM<br/>`goods` GROUP BY `g_cat`</pre> | inner_sum<br>500<br>1200 |

## 6.5.8. Анализ ошибок в подзапросах, оптимизация подзапросов и преобразование подзапросов в запросы с JOIN

### Анализ ошибок в подзапросах

Ошибка 1: неподдерживаемый синтаксис подзапросов.

```
ERROR 1235 (ER_NOT_SUPPORTED_YET)
```

Как правило, возникает в случае, если в подзапросе реализовано что-то, что (пока?) MySQL не умеет выполнять, например:

```
SELECT * FROM `t1` WHERE `s1`  
IN (SELECT `s2` FROM `t2` ORDER BY `s1` LIMIT 1)
```

LIMIT в контексте IN не поддерживается.

Ошибка 2: неверное количество колонок.

```
ERROR 1241 (ER_OPERAND_COL)
```

Как правило, возникает в случае, если подзапрос возвращает набор данных (колонок) там, где ожидался возврат скалярного значения:

```
SELECT (SELECT `c1`, `c2` FROM `t2`) FROM `t1`
```

Так вообще писать не надо!

Ошибка 3: неверное количество рядов.

```
ERROR 1242 (ER_SUBSELECT_NO_1_ROW)
```

Как правило, возникает в случае, если подзапрос возвращает набор данных (рядов) там, где ожидался возврат одного ряда:

```
SELECT * FROM `t1` WHERE `c1` = (SELECT `c1` FROM `t2`)
```

Такой запрос должен выглядеть так:

```
SELECT * FROM `t1` WHERE `c1` =  
ANY (SELECT `c1` FROM `t2`)
```

Ошибка 4: неверная работа с таблицами.

```
Error 1093 (ER_UPDATE_TABLE_USED)
```

Как правило, возникает в случае, если внешний запрос и подзапрос работают с одной и той же таблицей причём так, что выполнение операции приводит к конфликту или «вечному циклу»:

```
UPDATE `t1` SET `c1` = (SELECT MAX(`c1`) FROM `t1`)
```

Операция обновления влияет на результат операции чтения, которая используется для определения значения операции обновления.

## Оптимизация подзапросов

Указывайте сортировку результатов подзапроса:

```
SELECT * FROM `t1` WHERE `t1`.`c1` IN  
(SELECT `c1` FROM `t2` ORDER BY `c1`)
```

Устраняйте дублирование результатов подзапроса:

```
SELECT * FROM `t1` WHERE `t1`.`c1` IN  
(SELECT DISTINCT `c1` FROM `t2`)
```

Сокращайте до минимума количество рядов в подзапросе:

```
SELECT * FROM `t1` WHERE EXISTS  
(SELECT * FROM `t2` LIMIT 1)
```

В некоторых случаях использование подзапросов оказывается более выгодным, чем использование JOIN. Например, такой запрос

```
SELECT DISTINCT `c1` FROM `t1` WHERE `t1`.`c1`  
IN (SELECT `c1` FROM `t2`)
```

будет работать быстрее, чем такой:

```
SELECT DISTINCT `t1`.`c1` FROM `t1`, `t2`  
WHERE `t1`.`c1` = `t2`.`c1`
```

Помещайте условия ВНУТРЬ подзапросов. Например, такой подзапрос будет работать быстрее

```
SELECT * FROM `t1` WHERE `c1` IN (SELECT `c1` FROM `t1`  
UNION ALL SELECT `c1` FROM `t2`)
```

чем такой:

```
SELECT * FROM `t1` WHERE `c1` IN (SELECT `c1` FROM `t1`)  
OR `c1` IN (SELECT `c1` FROM `t2`)
```

Используйте подзапрос, возвращающий ряд, вместо взаимосвязанных запросов и подзапросов. Например, такой запрос будет работать быстрее

```
SELECT * FROM `t1` WHERE (`c1`, `c2`)  
IN (SELECT `c1`, `c2` FROM `t2`)
```

чем такой:

```
SELECT * FROM `t1` WHERE EXISTS  
(SELECT * FROM `t2` WHERE  
`t2`.`c1` = `t1`.`c1` AND `t2`.`c2` = `t1`.`c2`)
```

На этом множество идей по оптимизации подзапросов не заканчивается.

См. подробности здесь:

<http://dev.mysql.com/doc/refman/5.5/en/optimizing-subqueries.html>

Важно помнить, что эти (и другие) советы по оптимизации даны «для общего случая», и в вашей конкретной ситуации могут даже снизить производительность. Поэтому – проводите исследование на реальных данных в реальной базе данных.

## Преобразование подзапросов в запросы с JOIN

Ранее уже было сказано, что иногда подзапросы эффективнее запросов с JOIN. Но иногда – наоборот, запросы с JOIN оказываются более выгодными по тем или иным критериям.

Рассмотрим пару случаев преобразования подзапросов в запросы с JOIN...

Самый частый случай – замена IN и подзапроса, т.е. вместо

```
SELECT * FROM `t1` WHERE `id` IN  
(SELECT `id` FROM `t2`)
```

МОЖНО написать:

```
SELECT DISTINCT `t1`.* FROM `t1`, `t2`  
WHERE `t1`.`id` = `t2`.`id`
```

ИЛИ

```
SELECT DISTINCT `t1`.* FROM `t1` JOIN `t2` USING(`id`)
```

Ещё один случай – использование внешних объединений вместо IN или EXISTS. Например эти два запроса

```
SELECT * FROM `t1` WHERE `id` NOT IN  
(SELECT `id` FROM `t2`)
```

```
SELECT * FROM `t1` WHERE NOT EXISTS  
(SELECT `id` FROM `t2` WHERE `t1`.`id` = `t2`.`id`)
```

МОЖНО записать так:

```
SELECT `t1`.* FROM `t1` LEFT JOIN `t2`  
ON `t1`.`id` = `t2`.`id` WHERE `t2`.`id` IS NULL
```

### 6.5.9. Промежуточный итог

Мы рассмотрели большую часть материала, связанного с управлением данными.

В следующих разделах многие фрагменты синтаксиса операторов будут эквивалентны тому, что было рассмотрено в разделах «Оператор SELECT» и «Подзапросы», потому мы просто будем ссылаться на уже изученный материал.

## 6.6. Оператор UPDATE

### 6.6.1. Общая структура оператора UPDATE

Фрагмент table\_reference(s) в операторе UPDATE эквивалентен аналогичному фрагменту оператора SELECT, потому здесь мы не будем его рассматривать.

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}] ...
[WHERE where_condition]
[ORDER BY ...]
[LIMIT row_count]
```

```
UPDATE [LOW_PRIORITY] [IGNORE] table_references
SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}] ...
[WHERE where_condition]
```

– синтаксис для работы с одной таблицей.

– синтаксис для работы с несколькими таблицами.

### 6.6.2. UPDATE: [LOW\_PRIORITY]

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}] ...
[WHERE where_condition]
[ORDER BY ...]
[LIMIT row_count]
```

```
UPDATE [LOW_PRIORITY] [IGNORE] table_references
SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}] ...
[WHERE where_condition]
```

Использование LOW\_PRIORITY позволяет отложить выполнение обновления, пока с затронутыми таблицами выполняются операции чтения.

Это актуально только для методов доступа, выполняющих блокировку на уровне таблиц: MyISAM, MEMORY, MERGE.

### 6.6.3. UPDATE: [IGNORE]

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}] ...
[WHERE where_condition]
[ORDER BY ...]
[LIMIT row_count]
```

```
UPDATE [LOW_PRIORITY] [IGNORE] table_references
SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}] ...
[WHERE where_condition]
```

Использование IGNORE позволяет игнорировать два вида ошибок:

- дупликация первичных ключей (запись не обновляется);
- небезопасная конвертация данных (данные обновляются до ближайшего корректного значения).

## 6.6.4. UPDATE: SET

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
  SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}] ...
  [WHERE where_condition]
  [ORDER BY ...]
  [LIMIT row_count]
```

```
UPDATE [LOW_PRIORITY] [IGNORE] table_references
  SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}] ...
  [WHERE where_condition]
```

Основная часть оператора, в которой указывается, какие значения следует присвоить тому или иному полю таблицы.

Использование ключевого слова DEFAULT допустимо только в том случае, если у поля есть значение по умолчанию.

**Q:** А если значение по умолчанию не выставлено, но мы напишем UPDATE IGNORE ... SET `c` = DEFAULT ?

**A:** Сработает. В качестве «значения по умолчанию» будет выставлено «некое значение».

**Q:** Но ведь эти «некие значения» где-то описаны?!

**A:** Да. Вы на 100% уверены, что запомните их, и в любой версии СУБД они будут одинаковыми? Вот и не делайте так.

Рассмотрим пару простых примеров использования SET.

Пример 1. Изменить имя файла в галерее изображений.

```
UPDATE `pics` SET `p_file` = '5.jpg' WHERE `p_id` =2
```

Пример 2. Увеличить на 20% цену указанных товаров.

```
UPDATE `goods` SET `g_price` = `g_price` + `g_price` * 0.2
WHERE `g_id` IN (2, 5, 6)
```

Если бы цена была представлена в виде целого числа, то запрос выглядел бы так:

```
UPDATE `goods` SET `g_price` =
CONVERT( ROUND(`g_price` + `g_price` * 0.2, 0), UNSIGNED)
WHERE `g_id` IN (2, 5, 6)
```

## 6.6.5. UPDATE: [WHERE where\_condition]

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}] ...
[WHERE where_condition]
[ORDER BY ...]
[LIMIT row_count]
```

```
UPDATE [LOW_PRIORITY] [IGNORE] table_references
SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}] ...
[WHERE where_condition]
```

Эта часть ОБЯЗАТЕЛЬНО должна присутствовать, если только вы не хотите обновить ВСЕ записи в таблице. Подробнее про WHERE см. в разделе «Оператор SELECT».

**Q:** А можно ли здесь использовать HAVING?

**A:** Нет. (HAVING работает «после» выполнения запроса.)

## 6.6.6. UPDATE: [ORDER BY ...]

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}] ...
[WHERE where_condition]
[ORDER BY ...]
[LIMIT row_count]
```

```
UPDATE [LOW_PRIORITY] [IGNORE] table_references
SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}] ...
[WHERE where_condition]
```

Управляет последовательностью обновления записей, что может быть полезно для исключения ситуации дублирования ключей или уникальных индексов.

Рассмотрим подробнее. Допустим, у нас есть такая таблица, `p\_unique\_value` – уникальный индекс, который надо «сместить на 100».



| Исходный вид таблицы |                |         |
|----------------------|----------------|---------|
| p_id                 | p_unique_value | p_data  |
| 1                    | 100            | V_GUEST |
| 2                    | 200            | V_USER  |
| 3                    | 300            | V_ADMIN |
| 4                    | 400            | V_ROOT  |
| 5                    | 500            | V_GOD   |

| Желаемый вид таблицы |                |         |
|----------------------|----------------|---------|
| p_id                 | p_unique_value | p_data  |
| 1                    | 200            | V_GUEST |
| 2                    | 300            | V_USER  |
| 3                    | 400            | V_ADMIN |
| 4                    | 500            | V_ROOT  |
| 5                    | 600            | V_GOD   |

Такой запрос не сработает (дупликация индекса):

```
UPDATE `properties` SET `p_unique_value` = `p_unique_value` + 100
```

А такой – сработает:

```
UPDATE `properties` SET `p_unique_value` = `p_unique_value` + 100 ORDER BY `p_unique_value` DESC
```

### 6.6.7. UPDATE: [LIMIT row\_count]

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}] ...
[WHERE where_condition]
[ORDER BY ...]
[LIMIT row_count]
```

```
UPDATE [LOW_PRIORITY] [IGNORE] table_references
SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}] ...
[WHERE where_condition]
```

Использование LIMIT приводит к тому, что операция обновления прекращается после того, как MySQL «отсчитает» LIMIT рядов, соответствующих WHERE (вне зависимости от того, были ли они на самом деле обновлены).

### 6.6.8. UPDATE: работа с несколькими таблицами

**Q:** А зачем нужна возможность обновлять сразу несколько таблиц, т.е. зачем нужен вот такой синтаксис?

```
UPDATE [LOW_PRIORITY] [IGNORE] table_references
SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}] ...
[WHERE where_condition]
```

**A:** Сразу несколько – и не надо. Просто в руководстве так «криво» описали возможность использования UPDATE вместе с JOIN. Имеется в виду, что обращаясь вы будете к нескольким таблицам, а обновлять одну. Да, можно и обновлять несколько, но, как правило, подобная задача решается проще.

Да, ешё! В этом синтаксисе НЕТ ни LIMIT, ни ORDER BY!

**Q:** А можно это как-то «на пальцах»?

**A:** Можно.

Вспомним уже рассмотренные ранее таблицы `rooms` и `comps`. В таблицу `rooms` добавим поле `r\_comps\_count` и обновим его до актуального состояния.

`rooms`

| r_id | r_name           | r_comps_count |
|------|------------------|---------------|
| 1    | RoomWithTwoComps | 0             |
| 2    | RoomWithOneComp  | 0             |
| 3    | EmptyRoomOne     | 0             |
| 4    | EmptyRoomTwo     | 0             |

`comps`

| c_id | r_id | c_name              |
|------|------|---------------------|
| 1    | 1    | FirstCompInRoomOne  |
| 2    | 1    | SecondCompInRoomOne |
| 3    | 2    | SingleCompInRoomTwo |
| 4    | NULL | FreeCompOne         |
| 5    | NULL | FreeCompTwo         |

```
UPDATE `rooms` LEFT OUTER JOIN
(SELECT `r_id`, COUNT(*) AS `x` FROM `comps` GROUP BY `r_id`)
AS `C` USING (`r_id`) SET `rooms`.`r_comps_count` = `x`
```

Отметим, что эта же задача решается и гораздо более простым способом, т.е. вместо

```
UPDATE `rooms` LEFT OUTER JOIN
(SELECT `r_id`, COUNT(*) AS `X` FROM `comps` GROUP BY `r_id`)
AS `C` USING (`r_id`) SET `rooms`.`r_comps_count` = `X`
```

МОЖНО написать:

```
UPDATE `rooms` SET `r_comps_count` = (SELECT COUNT(*) FROM `comps` WHERE
`comps`.`r_id` = `rooms`.`r_id`)
```

Результат выполнения запросов – здесь.

| Запрос  | Результат   |               |        |               |   |                  |   |   |                 |   |   |              |      |   |              |      |
|---|---|---------------|--------|---------------|---|------------------|---|---|-----------------|---|---|--------------|------|---|--------------|------|
| <pre>UPDATE `rooms` LEFT OUTER JOIN (SELECT `r_id`, COUNT(*) AS `X` FROM `comps` GROUP BY `r_id`) AS `C` USING (`r_id`) SET `rooms`.`r_comps_count` = `X`</pre> | <table border="1"> <thead> <tr> <th>r_id</th><th>r_name</th><th>r_comps_count</th></tr> </thead> <tbody> <tr> <td>1</td><td>RoomWithTwoComps</td><td>2</td></tr> <tr> <td>2</td><td>RoomWithOneComp</td><td>1</td></tr> <tr> <td>3</td><td>EmptyRoomOne</td><td>NULL</td></tr> <tr> <td>4</td><td>EmptyRoomTwo</td><td>NULL</td></tr> </tbody> </table> | r_id          | r_name | r_comps_count | 1 | RoomWithTwoComps | 2 | 2 | RoomWithOneComp | 1 | 3 | EmptyRoomOne | NULL | 4 | EmptyRoomTwo | NULL |
| r_id  | r_name  | r_comps_count |        |               |   |                  |   |   |                 |   |   |              |      |   |              |      |
| 1   | RoomWithTwoComps  | 2             |        |               |   |                  |   |   |                 |   |   |              |      |   |              |      |
| 2   | RoomWithOneComp   | 1             |        |               |   |                  |   |   |                 |   |   |              |      |   |              |      |
| 3   | EmptyRoomOne  | NULL          |        |               |   |                  |   |   |                 |   |   |              |      |   |              |      |
| 4   | EmptyRoomTwo  | NULL          |        |               |   |                  |   |   |                 |   |   |              |      |   |              |      |
| <pre>UPDATE `rooms` SET `r_comps_count` = (SELECT COUNT(*) FROM `comps` WHERE `comps`.`r_id` = `rooms`.`r_id`)</pre>  | <table border="1"> <thead> <tr> <th>r_id</th><th>r_name</th><th>r_comps_count</th></tr> </thead> <tbody> <tr> <td>1</td><td>RoomWithTwoComps</td><td>2</td></tr> <tr> <td>2</td><td>RoomWithOneComp</td><td>1</td></tr> <tr> <td>3</td><td>EmptyRoomOne</td><td>0</td></tr> <tr> <td>4</td><td>EmptyRoomTwo</td><td>0</td></tr> </tbody> </table>       | r_id          | r_name | r_comps_count | 1 | RoomWithTwoComps | 2 | 2 | RoomWithOneComp | 1 | 3 | EmptyRoomOne | 0    | 4 | EmptyRoomTwo | 0    |
| r_id  | r_name  | r_comps_count |        |               |   |                  |   |   |                 |   |   |              |      |   |              |      |
| 1   | RoomWithTwoComps  | 2             |        |               |   |                  |   |   |                 |   |   |              |      |   |              |      |
| 2   | RoomWithOneComp   | 1             |        |               |   |                  |   |   |                 |   |   |              |      |   |              |      |
| 3   | EmptyRoomOne  | 0             |        |               |   |                  |   |   |                 |   |   |              |      |   |              |      |
| 4   | EmptyRoomTwo  | 0             |        |               |   |                  |   |   |                 |   |   |              |      |   |              |      |

## 6.7. Оператор INSERT

### 6.7.1. Общая структура оператора INSERT

Как и в случае с оператором UPDATE, мы не будем рассматривать подробно фрагменты, эквивалентные аналогичным фрагментам оператора SELECT.

Оператор INSERT поддерживает три варианта синтаксиса:

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name [(col_name,...)]
{VALUES | VALUE} ({expr | DEFAULT},...),(...),...
[ ON DUPLICATE KEY UPDATE
  col_name=expr
  [, col_name=expr] ... ]
```

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name
SET col_name={expr | DEFAULT}, ...
[ ON DUPLICATE KEY UPDATE
  col_name=expr
  [, col_name=expr] ... ]
```

См. SELECT.

```
INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name [(col_name,...)]
SELECT ...
[ ON DUPLICATE KEY UPDATE
  col_name=expr
  [, col_name=expr] ... ]
```

### 6.7.2. Классический вариант синтаксиса INSERT

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name [(col_name,...)]
{VALUES | VALUE} ({expr | DEFAULT},...),(...),...
[ ON DUPLICATE KEY UPDATE
  col_name=expr
  [, col_name=expr] ... ]
```

Данный вариант оператора INSERT является наиболее часто используемым, «классическим», потому основную информацию об INSERT мы будем рассматривать именно на примере этого синтаксиса.

### 6.7.3. INSERT: DELAYED

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name [(col_name,...)]
[VALUES | VALUE] ({expr | DEFAULT},...),(...),...
[ ON DUPLICATE KEY UPDATE
  col_name=expr
  [, col_name=expr] ... ]
```

Ключевое слово **DELAYED** позволяет реализовать поведение, при котором добавляемые записи помещаются в буфер на стороне сервера, а клиент может немедленно продолжить работу, не дожидаясь фактического завершения запроса.

Записи переносятся из буфера в таблицу в моменты, когда отсутствуют конфликты, вызванные блокировкой таблицы другими операциями.

См. подробности здесь:

<http://dev.mysql.com/doc/refman/5.5/en/insert-delayed.html>

### 6.7.4. INSERT: [INTO] ...

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name [(col_name,...)]
[VALUES | VALUE] ({expr | DEFAULT},...),(...),...
[ ON DUPLICATE KEY UPDATE
  col_name=expr
  [, col_name=expr] ... ]
```

Это – основная часть оператора **INSERT**, указывающая поля, в которые производится вставка, и соответствующие значения.

Примечание: слова **VALUE** и **VALUES** – синонимы.

Рассмотрим пример – добавим в таблицу товаров два тестовых товара.

| Таблица до вставки |       |            |         |            |       | Таблица после вставки |       |            |         |            |       |
|--------------------|-------|------------|---------|------------|-------|-----------------------|-------|------------|---------|------------|-------|
| g_id               | g_cat | g_name     | g_price | g_quantity | g_top | g_id                  | g_cat | g_name     | g_price | g_quantity | g_top |
| 1                  | 1     | i7         | 250     | 5          | 0     | 1                     | 1     | i7         | 250     | 5          | 0     |
| 2                  | 1     | Athlon     | 518.4   | 10         | 1     | 2                     | 1     | Athlon     | 518.4   | 10         | 1     |
| 3                  | 2     | HP1010     | 300     | 5          | 1     | 3                     | 2     | HP1010     | 300     | 5          | 1     |
| 4                  | 2     | HP2020     | 400     | 7          | 0     | 4                     | 2     | HP2020     | 400     | 7          | 0     |
| 5                  | 2     | Epson-N999 | 1036.8  | 9          | 1     | 5                     | 2     | Epson-N999 | 1036.8  | 9          | 1     |
|                    |       |            |         |            |       | 6                     | 1     | Test1      | 700     | 10         | 0     |
|                    |       |            |         |            |       | 7                     | 2     | Test2      | 800     | 20         | 0     |

Запрос

```
INSERT INTO `goods`
(`g_id` , `g_cat` , `g_name` , `g_price` , `g_quantity` , `g_top`)
VALUES
(NULL , '1' , 'Test1' , '700' , '10' , b'0' ) ,
(NULL , '2' , 'Test2' , '800' , '20' , b'0' )
```

Можно добавлять и одну строчку ☺.

Автоинкрементируемый РК.

Поле типа BIT.

**Q:** Только что мы использовали синтаксис INSERT, где за один раз добавляли несколько строк. А зачем? И сколько всего строк можно добавить за раз?

**A:** Вставка нескольких строк одной командой работает быстрее. Всего за один раз можно вставить столько строк, сколько «влазит в длину запроса». Никаких ограничений по количеству добавляемых строк (или значений) нет.

Единственное ограничение – значение max\_allowed\_packet, см. подробности здесь:

[http://dev.mysql.com/doc/refman/5.1/en/server-system-variables.html#sysvar\\_max\\_allowed\\_packet](http://dev.mysql.com/doc/refman/5.1/en/server-system-variables.html#sysvar_max_allowed_packet)

### 6.7.5. INSERT: ON DUPLICATE KEY UPDATE

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name [(col_name,...)]
[VALUES | VALUE] ({expr | DEFAULT},...),(...),
[ ON DUPLICATE KEY UPDATE
  col_name=expr
  [, col_name=expr] ... ]
```

Данная секция описывает поведение СУБД в случае, если в процессе вставки данных произошло дублирование первичного ключа или уникального индекса.

Также рекомендуется почитать: <http://habrahabr.ru/post/156489/>

Пример: в таблицу товаров (PK(`g\_id`)) добавим товар так, чтобы это привело к дублированию РК.

| Исходная таблица |       |            |         |            |       |  | Запрос | Результат |
|------------------|-------|------------|---------|------------|-------|--|--------|-----------|
| g_id             | g_cat | g_name     | g_price | g_quantity | g_top |  |        |           |
| 1                | 1     | i7         | 250     | 5          | 0     |  |        |           |
| 2                | 1     | Athlon     | 518.4   | 10         | 1     |  |        |           |
| 3                | 2     | HP1010     | 300     | 5          | 1     |  |        |           |
| 4                | 2     | HP2020     | 400     | 7          | 0     |  |        |           |
| 5                | 2     | Epson-N999 | 1036.8  | 9          | 1     |  |        |           |
| 6                | 1     | Test1      | 700     | 10         | 0     |  |        |           |
| 7                | 2     | Test2      | 800     | 20         | 0     |  |        |           |

```
INSERT INTO `goods`
(`g_id` , `g_cat` ,
`g_name` , `g_price` ,
`g_quantity` , `g_top`)
VALUES
(1 , '1', 'Test1',
'700', '10', b'0')
ON DUPLICATE KEY UPDATE
`g_id` = `g_id` + 1
```

⚠ #1557 - Upholding foreign key constraints for table 'goods', entry 2', key 2 would lead to a duplicate entry

Т.е. попытка заставить MySQL циклически обновлять ключ, пока не будет устранено совпадение, не работает.

Это вполне логично, т.к. ON DUPLICATE KEY UPDATE предназначено для обновления неключевых полей в случае совпадения ключевых. Т.е. для выполнения запросов вида:

| Запрос  | Результат (фрагмент)  |         |         |            |         |            |       |   |   |         |    |   |   |
|---|---|---------|---------|------------|---------|------------|-------|---|---|---------|----|---|---|
|   | g_id g_cat g_name g_price g_quantity g_top  |         |         |            |         |            |       |   |   |         |    |   |   |
| <pre>INSERT INTO `goods` (`g_id`, `g_cat`, `g_name`, `g_price` , `g_quantity`, `g_top`) VALUES (1, '1', 'NewName', '10', '1', b'0') ON DUPLICATE KEY UPDATE `g_cat` = '1', `g_name` = 'NewName', `g_price` = 10, `g_quantity` = 1, `g_top` = b'0'</pre> | <table border="1"> <tr> <th>g_id</th><th>g_cat</th><th>g_name</th><th>g_price</th><th>g_quantity</th><th>g_top</th></tr> <tr> <td>1</td><td>1</td><td>NewName</td><td>10</td><td>1</td><td>0</td></tr> </table> | g_id    | g_cat   | g_name     | g_price | g_quantity | g_top | 1 | 1 | NewName | 10 | 1 | 0 |
| g_id  | g_cat   | g_name  | g_price | g_quantity | g_top   |            |       |   |   |         |    |   |   |
| 1   | 1   | NewName | 10      | 1          | 0       |            |       |   |   |         |    |   |   |

В любом случае, использование ON DUPLICATE KEY UPDATE не является типичным и рекомендованным. Для этого должны быть веские причины (слияние баз данных, восстановление инкрементальных резервных копий и т.п.), но и в таких случаях надо:

1. Внимательно изучить: <http://dev.mysql.com/doc/refman/5.5/en/insert-on-duplicate.html>
2. Поискать альтернативные решения, которые, возможно, окажутся более логичными и устойчивыми.

## 6.7.6. INSERT: [IGNORE]

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name [(col_name,...)]
[VALUES | VALUE] ((expr | DEFAULT),...),(...),...
[ ON DUPLICATE KEY UPDATE
  col_name=expr
  [, col_name=expr] ... ]
```

Как и в случае оператора UPDATE, опция IGNORE позволяет игнорировать ошибки (дупликация ключей, некорректные данные и т.п.), переводя их в разряд предупреждений.

Операция выполняется, но теперь очень сложно утверждать, что в базу данных было добавлено то и так, что и как мы хотели добавить.

## 6.7.7. Синтаксис INSERT в стиле UPDATE

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name
SET col_name=expr | DEFAULT}, ...
[ ON DUPLICATE KEY UPDATE
  col_name=expr
  [, col_name=expr] ... ]
```

Все фрагменты оператора нам уже знакомы. Этот вариант синтаксиса лишь предполагает явное указание пар ПОЛЕ=ЗНАЧЕНИЕ. Совместим ответ на вопрос и пример.

**Q:** А можно ли тут не указывать значение автоинкрементируемого первичного ключа?

**A:** Да, например:

```
INSERT INTO `goods` SET `g_cat`=1, `g_name`='ABC', `g_price`=999,
`g_quantity`=1, `g_top`=b'0'
```

## 6.7.8. Синтаксис INSERT ... SELECT

```
INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name [(col_name,...)]
SELECT ...
[ ON DUPLICATE KEY UPDATE
  col_name=expr
  [, col_name=expr] ... ]
```

Здесь тоже всё уже рассмотрено. Единственная особенность такого синтаксиса состоит в том, что источником данных является не набор переданных значений, а результат запроса.

Зачем это нужно?

Варианты использования INSERT ... SELECT:

1. Копирование (фрагмента) древовидной структуры.
2. Копирование данных с некоторыми изменениями.
3. Перенос данных из одной БД в другую.
4. Денормализация (в целях кэширования).
5. Агрегация данных.

Рассмотрим пример агрегации данных.

Допустим, в интернет-магазине ОЧЕНЬ много товаров и категорий товаров. И при этом необходимо постоянно выводить информацию вида: «Товаров: X. Категорий: Y».

Логично будет кэшировать эту информацию в некоторой таблице, которая всё равно читается постоянно в систему служебных целей.

Варианты решения:

1. Очень заманчиво, но не работает (происходит JOIN, который тут не нужен):

```
INSERT INTO `aggregation` (`ag_categories`, `ag_goods`)
SELECT COUNT(`goods`.`g_id`), COUNT(`categories`.`c_id`)
FROM `goods`, `categories`
```

2. Использовать хранимые подпрограммы. Но об этом – позже.

3. Использовать серию запросов, например, так:

```
SELECT COUNT(`goods`.`g_id`) INTO @goods FROM `goods`;
SELECT COUNT(`categories`.`c_id`) INTO @categories FROM `categories`;
INSERT INTO `aggregation` (`ag_categories`, `ag_goods`) VALUES (@categories,
@goods);
```

4. Использовать подзапросы:

```
INSERT INTO `aggregation` (`ag_categories`, `ag_goods`)
VALUES ((SELECT COUNT(`categories`.`c_id`) FROM `categories`),
(SELECT COUNT(`goods`.`g_id`) FROM `goods`))
```

| ag_categories | ag_goods |
|---------------|----------|
| 2             | 7        |

Ещё INSERT ... SELECT используется для кэширования данных путём денормализации. Например, для каких-то нетривиальных целей нам нужно свести в одну таблицу информацию обо всех комнатах и компьютерах. Сделать это можно так:

```
INSERT INTO `denormalisation` (`rooms`, `comps`)
(SELECT `r_name`, `c_name` FROM `rooms`
LEFT OUTER JOIN `comps` USING(`r_id`))
UNION DISTINCT
(SELECT `r_name`, `c_name` FROM `rooms`
RIGHT OUTER JOIN `comps` USING(`r_id`))
```

| rooms            | comps               |
|------------------|---------------------|
| RoomWithTwoComps | FirstCompInRoomOne  |
| RoomWithTwoComps | SecondCompInRoomOne |
| RoomWithOneComp  | SingleCompInRoomTwo |
| EmptyRoomOne     | NULL                |
| EmptyRoomTwo     | NULL                |
| NULL             | FreeCompOne         |
| NULL             | FreeCompTwo         |

### 6.7.9. INSERT: важное напоминание, вопросы и ответы

**ВАЖНО!** При всех операциях, связанных с обновлением или вставкой данных, надо ВНИМАТЕЛЬНО следить за кодировками. В частности, попытка вставить UTF8 данные в поле с кодировкой latin1 с некоторой вероятностью приводит к падению запроса, даже несмотря на использование mysql\_real\_escape\_string().

**Q:** Есть таблица, на которой построено много индексов. Надо вставить кучу данных. Тормозит. Как сделать, чтобы не тормозило?!

**A:** Вот так.

1. Отключить индексы (первичные ключи и уникальные индексы не отключаются!):

```
ALTER TABLE ... DISABLE KEYS
```

2. Вставить данные.

3. Включить индексы:

```
ALTER TABLE ... ENABLE KEYS
```

**Q:** Как перенести данные из одной базы данных к другой?

**A:** Нужно иметь доступ к обеим базам, а затем – всё просто:

```
INSERT INTO `db2`.`tbl1` ... SELECT ... FROM `db1`.`tbl1`
```

**Q:** Нужно ли брать в кавычки в запросах ключевые слова типа NULL, DEFAULT и т.д.?

**A:** НЕТ! Иначе СУБД посчитает их текстовыми константами!

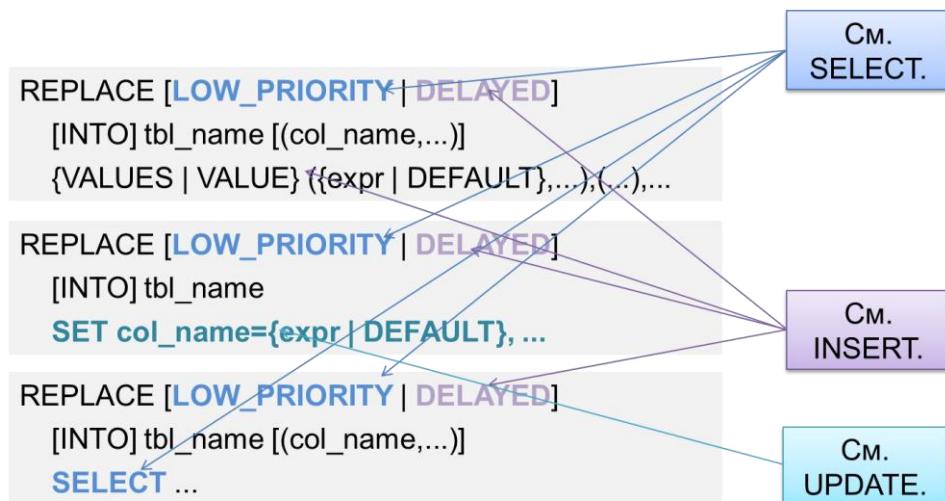


## 6.8. Оператор REPLACE

### 6.8.1. Общая структура и назначение оператора REPLACE

Продолжаем ту же логику, т.е. не рассматриваем повторно уже рассмотренное. В случае оператора REPLACE нам повезло – мы уже рассматривали все его фрагменты.

Тем не менее, краткий обзор структуры сделаем. Поддерживается три варианта синтаксиса:



При использовании оператора REPLACE СУБД проверяет первичные ключи и уникальные индексы на совпадение со значениями в новой записи:

- если есть совпадение – имеющаяся строка удаляется, а новая вставляется (при этом значение автоинкрементируемого ключа не увеличивается);
- если совпадений нет – просто вставляется новая строка.

### 6.8.2. Области применения и пример использования REPLACE

Допустим, у нас есть таблица-лог, в которой хранится время последнего обращения к ресурсу с некоторого ip-адреса. Здесь логично использовать именно REPLACE:

```
REPLACE `visits` SET `ip`='127.0.0.1'
```

| PK        |                     |
|-----------|---------------------|
| ip        | time                |
| 127.0.0.1 | 2012-11-28 11:50:34 |

Оператор REPLACE применяется в тех случаях, когда нам важно конечное состояние данных, и при том НЕ важно, были ли уже некие данные в таблице или они попадают туда впервые. Примеры реальных задач:

- Импорт данных с обновлением имеющихся.
- Показ сообщений в стиле «Вы посещали сайт последний раз с ip такого-то в такое-то время».
- Агрегация и кэширование, когда надо {либо впервые подготовить данные, либо обновить существующие}.

### 6.8.3. Области применения и пример использования REPLACE

**Q:** Как узнать, была ли замена (т.е. удаление и вставка) или вставка записи?

**А:** По количеству affected\_rows (2 или 1). Внимание! Если вы использовали REPLACE с 2+ наборами значений, этот способ «не работает», т.к. если вы получили, например, affected\_rows = 3 при передаче двух значений, то была одна вставка и одна замена, но какая запись была вставлена, а какая – заменена, выяснить уже нельзя. Впрочем, чаще всего, и не нужно ☺.

**Q:** Какие триггеры и в какой последовательности срабатывают при использовании REPLACE?

**А:** Такие и в такой:

- before insert
  - [before delete]
  - [after delete]
  - after insert

**Q:** Работает ли REPLACE быстрее, чем UPDATE?

**А:** Нет. Даже медленнее (т.к. происходит DELETE + INSERT).

**ВНИМАНИЕ!** Если вы пытаетесь обновить данные в родительской таблице, у которой проставлена каскадная операция удаления на дочернюю, возможны фатальные последствия!

Как только в родительской таблице произойдёт операция удаления записи, автоматически каскадно удалятся записи из дочерней таблицы.

#### **6.8.4. Краткий итог по REPLACE**

Чётко отдавайте себе отчёт в том, ЧТО вы хотите сделать:

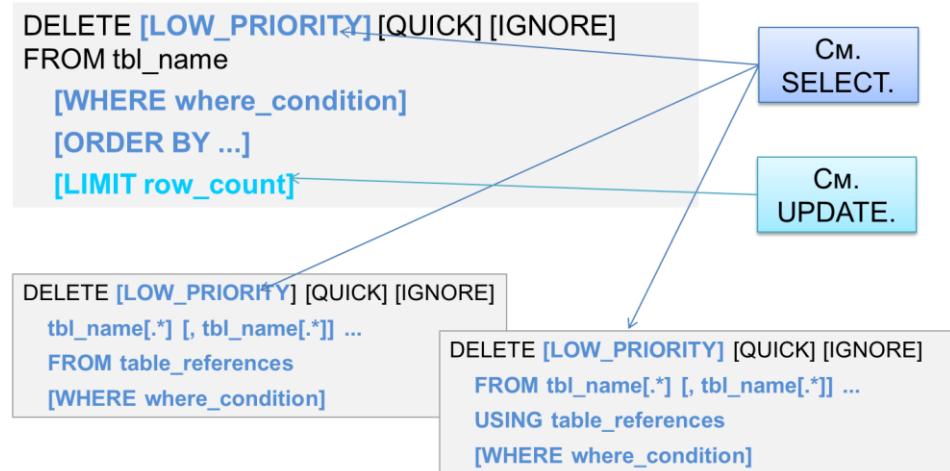
- Если вам надо гарантированно ДОБАВИТЬ строку – используйте INSERT.
  - Если вам нужно гарантированно ИЗМЕНИТЬ строку – используйте UPDATE.
  - Если вам нужно (в зависимости от наличия строки) ИЛИ добавить её, ИЛИ изменить – используйте REPLACE.

Помните, что REPLACE == DELETE + INSERT. Со всеми вытекающими последствиями!

## 6.9. Оператор DELETE

### 6.9.1. Общая структура оператора DELETE

Оператор DELETE поддерживает насколько вариантов синтаксиса. Для работы с одной таблицей и несколькими таблицами:



### 6.9.2. DELETE: [QUICK]

```

DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
FROM tbl_name
[WHERE where_condition]
[ORDER BY ...]
[LIMIT row_count]

```

Ключевое слово QUICK говорит СУБД, что после операции удаления не нужно оптимизировать «дырки в индексе». Это удобно в случае, если в скором времени на «пустое место» попадут новые записи с такими же значениями индексированных полей, как у удалённых.

Дополнительные подробности см. здесь:  
<http://dev.mysql.com/doc/refman/5.5/en/delete.html>

### 6.9.3. DELETE: [IGNORE]

```

DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
FROM tbl_name
[WHERE where_condition]
[ORDER BY ...]
[LIMIT row_count]

```

Как и в других операторах управления данными ключевое слово IGNORE «заставляет» СУБД переводить ошибки в разряд предупреждений.

Например, при использовании IGNORE успешно проходит запрос на удаление с запретом каскадного удаления (да, ничего не удаляется, но ошибки выполнения запроса не возникает).

#### 6.9.4. DELETE: пример и важное напоминание

Как и в случае UPDATE/REPLACE, при использовании DELETE обязательно надо указать условие, иначе вы просто «грохнете всю таблицу».

Пример использования DELETE:

| Исходное состояние таблицы |       |            | Запрос  |            | Конечное состояние таблицы |      |       |        |         |            |       |
|----------------------------|-------|------------|---------|------------|----------------------------|------|-------|--------|---------|------------|-------|
| g_id                       | g_cat | g_name     | g_price | g_quantity | g_top                      | g_id | g_cat | g_name | g_price | g_quantity | g_top |
| 1                          | 1     | NewName    | 10      | 1          | 0                          |      |       |        |         |            |       |
| 2                          | 1     | Athlon     | 518.4   | 10         | 1                          |      |       |        |         |            |       |
| 3                          | 2     | HP1010     | 300     | 5          | 1                          |      |       |        |         |            |       |
| 4                          | 2     | HP2020     | 400     | 7          | 0                          |      |       |        |         |            |       |
| 5                          | 2     | Epson-N999 | 1036.8  | 9          | 1                          |      |       |        |         |            |       |
| 6                          | 1     | Test1      | 700     | 10         | 0                          |      |       |        |         |            |       |
| 7                          | 2     | Test2      | 800     | 20         | 0                          |      |       |        |         |            |       |

#### 6.9.5. DELETE: работа с несколькими таблицами

Q: Зачем в синтаксисе DELETE предусмотрена возможность работы с несколькими таблицами?

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
tbl_name.* [, tbl_name.*] ...
FROM table_references
[WHERE where_condition]
```

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
FROM tbl_name.* [,tbl_name.*] ...
USING table_references
[WHERE where_condition]
```

A: Логика здесь такая же, как и при работе с несколькими таблицами в контексте UPDATE. Используется это для того, чтобы «скрестить» DELETE и JOIN.

Рассмотрим пару областей применения.

Использовать DELETE при работе с несколькими таблицами можно для:

- Имитации каскадного удаления (осторожно! непривычный синтаксис!)

```
DELETE `goods`, `categories` FROM `goods`, `categories`
WHERE `goods`.`category` = `categories`.`id` AND `categories`.`id` = 1
```

- Удаления «связанных наборов значений» в случае связи М-2-М (например, удаляется группа товаров и все характеристики этих товаров).
- Поиска и удаления записей-дубликатов (по некоторым критериям) из разных таблиц (в т.ч. в разных БД).

#### 6.9.6. Способ быстрой полной очистки таблицы

Для быстрого очищения таблицы используется оператор TRUNCATE. В отличие от DELETE он:

- Не вызывает срабатывания delete-триггеров.
- Удаляет сразу весь набор данных, а не «запись за записью».
- Работает ГОРАЗДО быстрее.
- Не возвращает количество удалённых записей.
- Не обновляет (а просто очищает) индексы.

См. подробности здесь:

<http://dev.mysql.com/doc/refman/5.5/en/truncate-table.html>

## 6.10. Оператор LOAD DATA INFILE

### 6.10.1. Общая структура оператора LOAD DATA INFILE

Для лучшего понимания этого материала рекомендуется повторить SELECT ... INTO OUTFILE.

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
[REPLACE | IGNORE]
INTO TABLE tbl_name
[CHARACTER SET charset_name]
[FIELDS | COLUMNS]
[TERMINATED BY 'string']
[[OPTIONALLY] ENCLOSED BY 'char']
[ESCAPED BY 'char']
]
[LINES
[STARTING BY 'string']
[TERMINATED BY 'string']
]
[IGNORE number {LINES | ROWS}]
[(col_name_or_user_var,...)]
[SET col_name = expr,...]
```

См. SELECT.

Оператор LOAD DATA INFILE позволяет прочитать содержимое текстового файла в таблицу, причём выполняется это очень быстро (быстрее, чем чтение классических дампов БД).

Использование этого оператора – достаточно большая редкость, тем не менее – рассмотрим, как он работает. Итак.

### 6.10.2. LOAD DATA INFILE: [CONCURRENT]

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
[REPLACE | IGNORE]
INTO TABLE tbl_name
[CHARACTER SET charset_name]
[FIELDS | COLUMNS]
[TERMINATED BY 'string']
[[OPTIONALLY] ENCLOSED BY 'char']
[ESCAPED BY 'char']
]
[LINES
[STARTING BY 'string']
[TERMINATED BY 'string']
]
[IGNORE number {LINES | ROWS}]
[(col_name_or_user_var,...)]
[SET col_name = expr,...]
```

Ключевое слово CONCURRENT позволяет в некоторых случаях разрешить чтение из таблицы, в то время как LOAD DATA INFILE заполняет её данными. Это актуально для метода доступа MyISAM.

Подробности см. здесь:

<http://dev.mysql.com/doc/refman/5.5/en/load-data.html>

### 6.10.3. LOAD DATA INFILE: [CONCURRENT]

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
[REPLACE | IGNORE]
INTO TABLE tbl_name
[CHARACTER SET charset_name]
[FIELDS | COLUMNS]
[TERMINATED BY 'string']
[[OPTIONALLY] ENCLOSED BY 'char']
[ESCAPED BY 'char']
]
[LINES
[STARTING BY 'string']
[TERMINATED BY 'string']
]
[IGNORE number {LINES | ROWS}]
[[col_name_or_user_var,...]]
[SET col_name = expr,...]
```

Наличие ключевого слова LOCAL означает, что файл находится на стороне клиента. Отсутствие этого слова означает, что файл находится на стороне сервера. В обоих случаях есть ряд специфических ограничений.

В большинстве своём они интуитивно понятны, но лучше почитать подробнее: <http://dev.mysql.com/doc/refman/5.5/en/load-data.html>

### 6.10.4. LOAD DATA INFILE: [REPLACE | IGNORE]

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
[REPLACE | IGNORE]
INTO TABLE tbl_name
[CHARACTER SET charset_name]
[FIELDS | COLUMNS]
[TERMINATED BY 'string']
[[OPTIONALLY] ENCLOSED BY 'char']
[ESCAPED BY 'char']
]
[LINES
[STARTING BY 'string']
[TERMINATED BY 'string']
]
[IGNORE number {LINES | ROWS}]
[[col_name_or_user_var,...]]
[SET col_name = expr,...]
```

Эти ключевые слова описывают поведение СУБД в случае, если при вставке данных происходит дублирование РК или уникальных индексов.

- IGNORE (автоматически включается по умолчанию при использовании LOCAL) – игнорировать дубликаты.
- REPLACE – заменять записи в таблице дубликатами (+ см. «Оператор REPLACE»).

### 6.10.5. LOAD DATA INFILE: [CHARACTER SET charset\_name]

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
[REPLACE | IGNORE]
INTO TABLE tbl_name
[CHARACTER SET charset_name]
[FIELDS | COLUMNS]
[TERMINATED BY 'string']
[[OPTIONALLY] ENCLOSED BY 'char']
[ESCAPED BY 'char']
]
[LINES
[STARTING BY 'string']
[TERMINATED BY 'string']
]
[IGNORE number {LINES | ROWS}]
[[col_name_or_user_var,...]]
[SET col_name = expr,...]
```

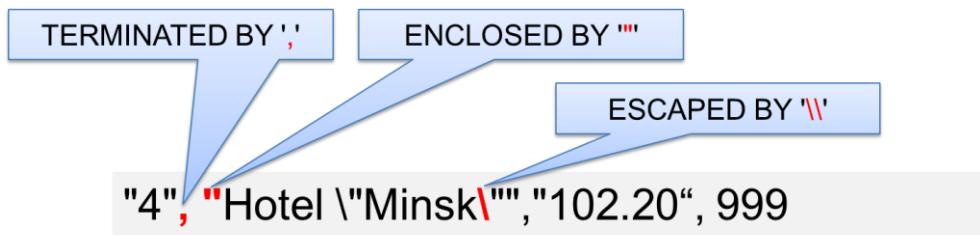
Позволяет указать кодировку, в которой находится текст в файле. Несмотря на то, что этот фрагмент является опциональным, **ВСЕГДА указывайте кодировку ЯВНО**, иначе может получиться много неприятных сюрпризов (даже тогда, когда кажется, что все умолчания указаны верно, всё везде совпадает и т.д.)

## 6.10.6. LOAD DATA INFILE: [FIELDS TERMINATED ... ENCLOSED ...]

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
[REPLACE | IGNORE]
INTO TABLE tbl_name
[CHARACTER SET charset_name]
[FIELDS | COLUMNS]
[TERMINATED BY 'string']
[[OPTIONALLY] ENCLOSED BY 'char']
[ESCAPED BY 'char']
]
[LINES
[STARTING BY 'string']
[TERMINATED BY 'string']
]
[IGNORE number {LINES | ROWS}]
[(col_name_or_user_var,...)]
[SET col_name = expr,...]
```

Всю эту секцию проще показать одной картинкой. Если хотите много-много текста – вам сюда:  
<http://dev.mysql.com/doc/refman/5.5/en/load-data.html>

Итак, пояснение картинкой:



## 6.10.7. LOAD DATA INFILE: [LINES STARTING ... TERMINATED BY ...]

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
[REPLACE | IGNORE]
INTO TABLE tbl_name
[CHARACTER SET charset_name]
[FIELDS | COLUMNS]
[TERMINATED BY 'string']
[[OPTIONALLY] ENCLOSED BY 'char']
[ESCAPED BY 'char']
]
[LINES
[STARTING BY 'string']
[TERMINATED BY 'string']
]
[IGNORE number {LINES | ROWS}]
[(col_name_or_user_var,...)]
[SET col_name = expr,...]
```

В некоторых случаях бывает необходимо указать последовательности символов, которые нужно трактовать как начало и конец анализируемых строк. Данный фрагмент оператора LOAD DATA INFILE позволяет указать эти ограничители.

Подробнее посмотрим на примере чуть позже.

## 6.10.8. LOAD DATA INFILE: [IGNORE number {LINES|ROWS}]

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'  
[REPLACE | IGNORE]  
INTO TABLE tbl_name  
[CHARACTER SET charset_name]  
[(FIELDS | COLUMNS)  
[TERMINATED BY 'string']  
[[OPTIONALLY] ENCLOSED BY 'char']  
[ESCAPED BY 'char']  
]  
[LINES  
[STARTING BY 'string']  
[TERMINATED BY 'string']  
]  
[IGNORE number {LINES | ROWS}]  
[(col_name_or_user_var,...)]  
[SET col_name = expr,...]
```

Самая простая часть оператора – указывает, сколько первых строк из файла пропустить.

При этом пропускается «то, что строкой посчитает СУБД» (см. LINES STARTING BY ... TERMINATED BY), а не то, что строкой посчитает человек, просматривающий файл в текстовом редакторе.

## 6.10.9. LOAD DATA INFILE: [SET col\_name = expr]

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'  
[REPLACE | IGNORE]  
INTO TABLE tbl_name  
[CHARACTER SET charset_name]  
[(FIELDS | COLUMNS)  
[TERMINATED BY 'string']  
[[OPTIONALLY] ENCLOSED BY 'char']  
[ESCAPED BY 'char']  
]  
[LINES  
[STARTING BY 'string']  
[TERMINATED BY 'string']  
]  
[IGNORE number {LINES | ROWS}]  
[(col_name_or_user_var,...)]  
[SET col_name = expr,...]
```

Позволяет выставлять тому или иному полю таблицы значение, отсутствующее в импортируемом файле, или полученное на основе уже импортированных (находящихся слева) колонок.

Это тоже будет в примере, который мы прямо сейчас и рассмотрим.

### 6.10.10. LOAD DATA INFILE: пример использования

Сначала создадим «текстовый дамп» вот такой таблицы:

| Код создания таблицы     |  | Данные в таблице |                     |                     |                  |                  |
|--------------------------|--|------------------|---------------------|---------------------|------------------|------------------|
|                          |  | <b>id</b>        | <b>string_field</b> | <b>double_field</b> | <b>bit_field</b> | <b>set_field</b> |
| CREATE TABLE `testtable` | (`id` int(11) NOT NULL AUTO_INCREMENT,<br>`string_field` varchar(255) DEFAULT NULL,<br>`double_field` double NOT NULL,<br>`bit_field` bit(1) NOT NULL,<br>`set_field` set('one','two') NOT NULL,<br>PRIMARY KEY (`id`)<br>) ENGINE=InnoDB DEFAULT CHARSET=utf8 | 1                | Hotel "Minsk"       | 11.22               | 1                | one              |
|                          |  | 2                | Hotel "Moscow"      | 44.55               | 0                | two              |
|                          |  | 3                | NULL                | 99.77               | 1                | one              |

Выполним запрос:

```
SELECT * INTO OUTFILE 'c:/testtable.txt' CHARACTER SET 'utf8'  
FIELDS TERMINATED BY ';' OPTIONALLY ENCLOSED BY '\"' ESCAPED BY '\\'  
LINES STARTING BY '+++' TERMINATED BY '---\n' FROM `testtable`
```

Получаем дамп (копия экрана специально сделана через FAR, чтобы точно видеть реальную разметку текста, значения и т.д.):

```
+++1;"Hotel \"Minsk\";11.22;@;"one"---  
+++2;"Hotel \"Moscow\";44.55; ;"two"---  
+++3;\N;99.77;@;"one"---
```

Теперь немного «покалечим» файл, очистим таблицу и прочитаем в неё данные из файла. При этом в таблицу добавим (в конец) ещё одно поле `new\_double`, значение которого будем получать как `double\_field`\*2.

| «Покалеченный файл»   | Новая структура таблицы  |                 |            |           |            |                |         |       |           |         |  |  |    |      |                |                     |              |                 |  |     |      |  |                     |        |  |  |    |      |  |                  |        |  |  |    |      |  |                  |                  |                 |  |    |      |  |                   |        |  |  |    |      |  |
|---|--|-----------------|------------|-----------|------------|----------------|---------|-------|-----------|---------|--|--|----|------|----------------|---------------------|--------------|-----------------|--|-----|------|--|---------------------|--------|--|--|----|------|--|------------------|--------|--|--|----|------|--|------------------|------------------|-----------------|--|----|------|--|-------------------|--------|--|--|----|------|--|
| <pre>+++1;"Hotel<br/>\"Minsk\";11.22;@;"one"---<br/>+++2;"Hotel \"<br/>Moscow\";44.55; ;"two"---<br/>+++3;\N;99.77;@;"one"---</pre> | <table border="1"> <thead> <tr> <th>Name</th><th>Type</th><th>Collation</th><th>Attributes</th><th>Null</th><th>Default</th><th>Extra</th></tr> </thead> <tbody> <tr> <td><b>id</b></td><td>int(11)</td><td></td><td></td><td>No</td><td>None</td><td>AUTO_INCREMENT</td></tr> <tr> <td><b>string_field</b></td><td>varchar(255)</td><td>utf8_general_ci</td><td></td><td>Yes</td><td>NULL</td><td></td></tr> <tr> <td><b>double_field</b></td><td>double</td><td></td><td></td><td>No</td><td>None</td><td></td></tr> <tr> <td><b>bit_field</b></td><td>bit(1)</td><td></td><td></td><td>No</td><td>None</td><td></td></tr> <tr> <td><b>set_field</b></td><td>set('one','two')</td><td>utf8_general_ci</td><td></td><td>No</td><td>None</td><td></td></tr> <tr> <td><b>new_double</b></td><td>double</td><td></td><td></td><td>No</td><td>None</td><td></td></tr> </tbody> </table> | Name            | Type       | Collation | Attributes | Null           | Default | Extra | <b>id</b> | int(11) |  |  | No | None | AUTO_INCREMENT | <b>string_field</b> | varchar(255) | utf8_general_ci |  | Yes | NULL |  | <b>double_field</b> | double |  |  | No | None |  | <b>bit_field</b> | bit(1) |  |  | No | None |  | <b>set_field</b> | set('one','two') | utf8_general_ci |  | No | None |  | <b>new_double</b> | double |  |  | No | None |  |
| Name  | Type   | Collation       | Attributes | Null      | Default    | Extra          |         |       |           |         |  |  |    |      |                |                     |              |                 |  |     |      |  |                     |        |  |  |    |      |  |                  |        |  |  |    |      |  |                  |                  |                 |  |    |      |  |                   |        |  |  |    |      |  |
| <b>id</b>   | int(11)  |                 |            | No        | None       | AUTO_INCREMENT |         |       |           |         |  |  |    |      |                |                     |              |                 |  |     |      |  |                     |        |  |  |    |      |  |                  |        |  |  |    |      |  |                  |                  |                 |  |    |      |  |                   |        |  |  |    |      |  |
| <b>string_field</b>   | varchar(255)   | utf8_general_ci |            | Yes       | NULL       |                |         |       |           |         |  |  |    |      |                |                     |              |                 |  |     |      |  |                     |        |  |  |    |      |  |                  |        |  |  |    |      |  |                  |                  |                 |  |    |      |  |                   |        |  |  |    |      |  |
| <b>double_field</b>   | double   |                 |            | No        | None       |                |         |       |           |         |  |  |    |      |                |                     |              |                 |  |     |      |  |                     |        |  |  |    |      |  |                  |        |  |  |    |      |  |                  |                  |                 |  |    |      |  |                   |        |  |  |    |      |  |
| <b>bit_field</b>  | bit(1)   |                 |            | No        | None       |                |         |       |           |         |  |  |    |      |                |                     |              |                 |  |     |      |  |                     |        |  |  |    |      |  |                  |        |  |  |    |      |  |                  |                  |                 |  |    |      |  |                   |        |  |  |    |      |  |
| <b>set_field</b>  | set('one','two')   | utf8_general_ci |            | No        | None       |                |         |       |           |         |  |  |    |      |                |                     |              |                 |  |     |      |  |                     |        |  |  |    |      |  |                  |        |  |  |    |      |  |                  |                  |                 |  |    |      |  |                   |        |  |  |    |      |  |
| <b>new_double</b>   | double   |                 |            | No        | None       |                |         |       |           |         |  |  |    |      |                |                     |              |                 |  |     |      |  |                     |        |  |  |    |      |  |                  |        |  |  |    |      |  |                  |                  |                 |  |    |      |  |                   |        |  |  |    |      |  |

Выполним запрос:

```
LOAD DATA INFILE 'c:/testtable.txt' INTO TABLE `testtable`  
CHARACTER SET 'utf8'  
FIELDS TERMINATED BY ';' OPTIONALLY ENCLOSED BY '\"' ESCAPED BY '\\'  
LINES STARTING BY '+++' TERMINATED BY '---\n'  
(`id`, `string_field`, `double_field`, `bit_field`, `set_field`) SET  
`new_double` = `double_field`*2
```

Получаем результат:

| <b>id</b> | <b>string_field</b> | <b>double_field</b> | <b>bit_field</b> | <b>set_field</b> | <b>new_double</b> |
|-----------|---------------------|---------------------|------------------|------------------|-------------------|
| 1         | Hotel<br>"Minsk"    | 11.22               | 1                | one              | 22.44             |
| 2         | Hotel "<br>Moscow"  | 44.55               | 0                | two              | 89.1              |
| 3         | NULL                | 99.77               | 1                | one              | 199.54            |

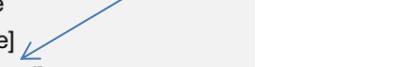
Всё! Рекомендую поэкспериментировать самостоятельно.

## 6.11. Оператор LOAD XML

### 6.11.1. Общая структура оператора LOAD XML

Мы только что рассматривали LOAD DATA INFILE. Единственное отличие LOAD XML в том, что работает он не с текстом, а с XML, поэтому здесь нет ключевых слов для управления разделителями, но есть возможность указать идентификатор ряда.

```
LOAD XML [LOW_PRIORITY | CONCURRENT] [LOCAL]
INFILE 'file_name'
[REPLACE | IGNORE]
INTO TABLE [db_name.]tbl_name
[CHARACTER SET charset_name]
[ROWS IDENTIFIED BY '<tagname>']
[IGNORE number {LINES | ROWS}]
[(column_or_user_var,...)]
[SET col_name = expr,...]
```



### 6.11.2. LOAD XML: пример использования

Возьмём результат работы LOAD DATA INFILE и «выгрузим в XML». Это удобно сделать через консоль.

```
mysql -uroot -p123456 --xml -e
"SELECT * FROM `shop`.`testtable`" > testtable.xml
```

Результат (фрагмент):

```
1  <?xml version="1.0"?>
2
3  <resultset statement="SELECT * FROM `shop`.`testtable`"
4  " xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
5    <row>
6      <field name="id">1</field>
7      <field name="string_field">Hotel
8
9      &quot;Minsk&quot;</field>
10     <field name="double_field">11.22</field>
11     <field name="bit_field"></field>
12     <field name="set_field">one</field>
13     <field name="new_double">22.44</field>
14   </row>
15
16   <row>
17     <field name="id">2</field>
18     <field name="string_field">Hotel &quot;
19
20
21     Moscow&quot;</field>
22     <field name="double_field">44.55</field>
23     <field name="bit_field"> </field>
24     <field name="set_field">two</field>
25     <field name="new_double">89.1</field>
26   </row>
```

Для чистоты эксперимента заменим тег `<row>` на `<one_row>` и проведём импорт данных.

```
LOAD XML INFILE 'c:/testtable.xml' IGNORE INTO TABLE `testtable`
CHARACTER SET 'UTF8' ROWS IDENTIFIED BY '<one_row>'
```

И всё, «приехали». Без IGNORE MySQL ругается на нулевые значения полей типа bit, но даже с IGNORE не может правильно обработать ситуацию со значением NULL в текстовом поле (считает, что поля нет).

В итоге данные оказываются повреждёнными.

|  | <input type="checkbox"/> Edit  Copy | <input type="checkbox"/> id | string_field | double_field | bit_field | set_field | new_double |
|--|-------------------------------------|-----------------------------|--------------|--------------|-----------|-----------|------------|
|  |                                     | 1                           | Hotel        | 11.22        | 1         | one       | 22.44      |
|  |                                     | 2                           | "Minsk"      | 44.55        | 1         | two       | 89.1       |
|  |                                     | 3                           | Hotel "      | 99.77        | 0         | one       | 199.54     |

И что делать? Ничего.

Для «классических случаев» (когда все поля у вас строковые или численные, нет NULL и т.п.) импорт из XML работает хорошо.

Про только что показанную проблему есть куча вопросов в Интернет (и ни одного разумительного ответа). Более-менее вменяемое написано здесь:

<http://stackoverflow.com/questions/8920995/load-xml-with-bit-fields-in-mysql>

### 6.11.3. LOAD XML: краткий итог

И ещё один важный момент.

Эксперименты с различными XML-файлами «разной степени покалеченности» показали, что довольно часто MySQL производит импорт, не выдаёт никаких сообщений об ошибках или предупреждений, но данные в таблице оказываются «повреждёнными» (т.е. отличаются от того, что было в файле).

Вывод? Ждём, пока этот функционал доведут до нормальной работоспособности.

## 6.12. Оператор CALL

### 6.12.1. Общая структура оператора CALL

Оператор CALL предназначен для вызова хранимых процедур. И всё ☺. Подробно рассматривать его будет в соответствующем разделе.

```
CALL sp_name([parameter[...]])  
CALL sp_name[()]
```

### 6.12.2. Пример хранимой процедуры и использования оператора CALL

А пока, чтобы составить общее представление, рассмотрим один пример.

Допустим, нам (по какой-то очень веской причине) нужно определить, у скольких записей первичный ключ лежит в заданном диапазоне, при этом нам известно только:

- Записи лежат в разных таблицах. Имена всех интересующих нас таблиц начинаются с "ctg".
- У каждой таблицы первичный ключ простой (состоит из одного поля), но его имя везде разное.

**ВАЖНО!** Если в реальной жизни у вас возникают подобные задачи, значит вы что-то очень и очень сильно проморгали на стадии разработки структуры базы данных

Итак, вот текст хранимой процедуры.

```
DROP PROCEDURE IF EXISTS RecordsCount;  
  
delimiter //  
  
CREATE PROCEDURE RecordsCount(IN PKmin INT, IN PKmax INT, OUT ResultCount INT)  
BEGIN  
  
-- Признак того, что в цикле перебора результатов запроса закончились ряды.  
DECLARE no_more_rows BOOLEAN DEFAULT FALSE;  
  
-- Переменная для хранения имени i-й перебираемое таблицы.  
DECLARE table_name CHAR(255) DEFAULT '';  
  
-- Счётчик количества найденных удовлетворяющих условию записей.  
DECLARE records_count INT DEFAULT 0;  
  
-- Создём курсор для запроса, который вернёт список таблиц,  
-- имена которых соответствуют шаблону.  
DECLARE tables_cursor CURSOR FOR  
    SELECT `information_schema`.`tables`.`TABLE_NAME`  
    FROM `information_schema`.`tables`  
    WHERE `information_schema`.`tables`.`TABLE_SCHEMA`='shop'  
    AND `information_schema`.`tables`.`TABLE_NAME` LIKE 'ctg\_%';  
  
-- Создаём обработчик ситуации "(больше) записей нет".  
DECLARE CONTINUE HANDLER FOR NOT FOUND SET no_more_rows = TRUE;  
  
-- Открываем курсор (т.е. выполняем запрос).  
OPEN tables_cursor;  
  
-- Сбрасываем счётчик.  
SET records_count = 0;
```

```
-- Цикл перебора всех найденных таблиц.
the_loop: LOOP

-- Извлекаем имя таблицы.
FETCH tables_cursor INTO table_name;

-- Формируем запрос, который вернёт имя первичного ключа таблицы.
SET @query = CONCAT("SELECT `information_schema`.`COLUMNS`.`COLUMN_NAME`"
                     INTO @tpk FROM `information_schema`.`COLUMNS`"
                     WHERE `information_schema`.`COLUMNS`.`TABLE_NAME` ='",
                     table_name, "' AND"
                     `information_schema`.`COLUMNS`.`COLUMN_KEY` = 'PRI'");

-- Формируем выполнимое выражение.
PREPARE real_query FROM @query;

-- Выполняем запрос. Теперь в переменной @tpk лежит имя PK таблицы.
EXECUTE real_query;

-- Формируем запрос, считающий количество записей, удовлетворяющих условию задачи
SET @query = CONCAT("SELECT COUNT(*) INTO @part FROM ", table_name, " WHERE `",
                     @tpk, "` >= ", PKmin, " AND `", @tpk, "` <= ", PKmax);

-- Формируем выполнимое выражение.
PREPARE real_query FROM @query;
-- Выполняем запрос. Теперь в переменной @part лежит
-- количество записей данной таблицы, удовлетворяющих условию задачи.
EXECUTE real_query;

-- Если мы перебрали все таблицы, чьи имена соответствуют шаблону,
-- выходим из цикла.
IF no_more_rows THEN
    CLOSE tables_cursor;
    LEAVE the_loop;
ELSE
    -- Накапливаем сумму количества записей по всем таблицам.
    SET records_count = records_count + @part;

    -- Это просто отладочный вывод.
    SELECT records_count;
END IF;

END LOOP the_loop;

SET ResultCount = records_count;

END
//  
delimiter ;
```

На уже известной нам базе данных интернет-магазина (см. ранее, был дан её дамп) выполним нашу процедуру.

```
CALL RecordsCount(1, 999, @a);
SELECT @a;
```

## Результат работы:

```
+-----+
| records_count |
+-----+
|          2   |
+-----+
1 row in set (0.09 sec)

+-----+
| records_count |
+-----+
|          5   |
+-----+
1 row in set (0.12 sec)

Query OK, 1 row affected (0.12 sec)

+-----+
| @a    |
+-----+
|      5  |
+-----+
1 row in set (0.00 sec)
```

При попытке вызвать процедуру из PhpMyAdmin, MySQL подвисает. Чтобы не подвисал, нужно убрать отладочный вывод внутри процедуры.

## 6.13. Оператор DO

### 6.13.1. Общая структура оператора DO

Оператор DO предназначен для выполнения любого допустимого SQL-выражения БЕЗ возврата результатов.

```
DO expr [, expr] ...
```

Это удобно при тестировании производительности. Сначала надо убедиться, что запрос выполняется верно, а затем заменить SELECT на DO, таким образом получив возможность «извне» замерять время непосредственного выполнения запроса, к которому не будет добавляться время извлечения данных.

Хотя, конечно, профилировщик тут более уместен.

## 6.14. Оператор HANDLER

### 6.14.1. Общая структура оператора HANDLER

Пожалуй, это самый редко используемый оператор доступа к данным, хотя в документации ему посвящён достаточно обширный раздел. Общий синтаксис этого оператора:

```
HANDLER tbl_name OPEN [ [AS] alias]

HANDLER tbl_name READ index_name { = | <= | >= | < | > } (value1,value2,...)
    [ WHERE where_condition ] [LIMIT ... ]
HANDLER tbl_name READ index_name { FIRST | NEXT | PREV | LAST }
    [ WHERE where_condition ] [LIMIT ... ]
HANDLER tbl_name READ { FIRST | NEXT }
    [ WHERE where_condition ] [LIMIT ... ]

HANDLER tbl_name CLOSE
```

### 6.14.2. Пример использования HANDLER

Чтобы не погружаться в экзотику, просто приведём пример того, как можно использовать HANDLER.

Каждый вызов READ возвращает одну запись.

```
HANDLER `goods` OPEN;

HANDLER `goods` READ FIRST; -- 1
HANDLER `goods` READ NEXT; -- 2
HANDLER `goods` READ NEXT; -- 3
HANDLER `goods` READ NEXT; -- 4
HANDLER `goods` READ NEXT; -- 5

HANDLER `goods` CLOSE;
```

### **6.14.3. Краткий итог по оператору HANDLER**

**Q:** Так а для чего он нужен-то!?

**А:** Этому вопросу посвящён почти весь раздел здесь:

<http://dev.mysql.com/doc/refman/5.5/en/handler.html>

Если в двух словах – это более быстрый вариант SELECT, но в силу КУЧИ ограничений и нюансов, преимущество в скорости кажется недостаточным для использования этого оператора.

Да, и ещё могут быть проблемы с клиентскими библиотеками, с помощью которых выполняются такие запросы. Вывод: используйте на свой страх и риск.

На этом мы завершаем рассмотрение языка управления данными (Data Manipulation Language, DML) в MySQL.

Для закрепления всего пройденного настоятельно рекомендуется позэкспериментировать. Особенно с оператором SELECT. Успехов!